UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Energy-aware scheduling in distributed computing systems

Santiago Iturriaga

Programa de Doctorado en Informática de PEDECIBA
Facultad de Ingeniería
Universidad de la República

Montevideo – Uruguay
Setiembre de 2017

UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Energy-aware scheduling in distributed computing systems

Santiago Iturriaga

Tesis de Doctorado presentada al Programa de Posgrado en Informática de PEDECIBA, Facultad de Ingeniería, Universidad de la República, como parte de los requisitos necesarios para la obtención del título de Doctor en Informática de PEDECIBA.

Directores de tesis:
Sergio Nesmachnow
Bernabé Dorronsoro

Director académico:
Sergio Nesmachnow

Montevideo – Uruguay
Setiembre de 2017

Iturriaga, Santiago

Energy-aware scheduling in distributed computing systems / Santiago Iturriaga. - Montevideo: Universidad de la República, Facultad de Ingeniería, 2017.

XV, 162 p.: il.; 29, 7cm.

Directores de tesis:

Sergio Nesmachnow

Bernabé Dorronsoro

Director académico:

Sergio Nesmachnow

Tesis de Doctorado – Universidad de la República, Programa en Informática de PEDECIBA, 2017.

Referencias bibliográficas: p. 141 – 162.

1. centro de datos, 2. eficiencia energética, 3. planificación de tareas.

INTEGRANTES DEL TRIBUNAL DE DEFENSA DE TESIS

Pascal Bouvry

Grégoire Danoy, revisor

Juan José Durillo, revisor

Pablo Monzón

Franco Robledo, presidente

Montevideo – Uruguay
Setiembre de 2017

Agradecimientos

Agradezco a Sergio por haberme guiado durante todo mi doctorado. Su permanente e insistente estimulo me motivó a seguir adelante en todo momento. Sergio ha estado fuertemente involucrado y me ha guiado en todos los detalles de la tesis, desde la concepción de sus objetivos, pasando por la investigación misma, y hasta la redacción de los artículos resultado de esta investigación. Sus aportes han sido invalorables.

Agradezco también a Bernabé por su confianza y apoyo. A pesar de las distancias, Bernabé supo estar presente en instancias claves de mi doctorado y realizó valiosos aportes a mi formación.

También agradezco al Instituto de Computación por el apoyo que me brindó y en particular a mis compañeros por cubrirme durante mis estancias de investigación en el exterior. Agradezco a la Comisión Académica de Posgrado, la Comisión Sectorial de Investigación Científica, la Agencia Nacional de Investigación e Innovación, y el Programa de Desarrollo de las Ciencias Básicas por sus apoyos económicos sin los cuales no me hubiera sido posible concluir mis estudios.

Finalmente, un agradecimiento aparte merece mi familia. Mi madre, mi padre y mi hermano que me apoyaron en todo momento, y Caro que soportó estos dos últimos años de trabajo y me acompañó en todo momento, aún durante estancias de investigación en el exterior. A todos ellos dedico este trabajo.

RESUMEN

Los sistemas informáticos distribuidos, como los centros de datos, son clave para satisfacer la demanda informática moderna. Sin embargo, su consumo de energético se ha convertido en una gran preocupación. Se estima que mundialmente su consumo energético rondó los 270 TWh en el año 2012, y algunos prevén que este consumo se cuadruplicará para el año 2030. Maximizar simultáneamente la eficiencia energética y computacional de los centros de datos es un desafío crítico. Esta tesis aborda dicho desafío mediante la planificación de la operativa del centro de datos considerando un enfoque multiobjetivo para optimizar simultáneamente ambos objetivos de eficiencia.

En esta tesis se estudian múltiples variantes del problema, desde la planificación de un único centro de datos hasta la de una federación de múltiples centros de datos geográficamente distribuidos. Para esto, se formulan modelos matemáticos para cada variante del problema, modelado sus componentes más relevantes, como: recursos computacionales, carga de trabajo, refrigeración, redes, energía verde, etc. Para resolver el problema de planificación planteado, se diseñan un conjunto de algoritmos heurísticos y metaheurísticos. Estos son estudiados exhaustivamente y su eficiencia es evaluada utilizando una batería de herramientas estadísticas.

Los resultados experimentales muestran que los algoritmos de planificación diseñados son capaces de aumentar significativamente la eficiencia energética de un centros de datos en comparación con métodos tradicionales planificación. A su vez, los métodos propuestos proporcionan un conjunto diverso de soluciones con diferente nivel de compromiso respecto a la eficiencia computacional del centro de datos. Estos resultados confirman la eficacia del enfoque algorítmico propuesto.

Palabras claves:
centro de datos, eficiencia energética, planificación de tareas.

ABSTRACT

Distributed computing systems, such as data centers, are key for supporting modern computing demands. However, the energy consumption of data centers has become a major concern over the last decade. Worldwide energy consumption in 2012 was estimated to be around 270 TWh, and grim forecasts predict it will quadruple by 2030. Maximizing energy efficiency while also maximizing computing efficiency is a major challenge for modern data centers. This work addresses this challenge by scheduling the operation of modern data centers, considering a multi-objective approach for simultaneously optimizing both efficiency objectives.

Multiple data center scenarios are studied, such as scheduling a single data center and scheduling a federation of several geographically-distributed data centers. Mathematical models are formulated for each scenario, considering the modeling of their most relevant components such as computing resources, computing workload, cooling system, networking, and green energy generators, among others. A set of accurate heuristic and metaheuristic algorithms are designed for addressing the scheduling problem. These scheduling algorithms are comprehensively studied, and compared with each other, using statistical tools to evaluate their efficacy when addressing realistic workloads and scenarios.

Experimental results show the designed scheduling algorithms are able to significantly increase the energy efficiency of data centers when compared to traditional scheduling methods, while providing a diverse set of trade-off solutions regarding the computing efficiency of the data center. These results confirm the effectiveness of the proposed algorithmic approaches for data center infrastructures.

Keywords:
data centers, energy efficiency, job scheduling.

Preface

The main contributions of this thesis are based on the following publications.

- Iturriaga, S., Dorronsoro, B., and Nesmachnow, S. (2017). Multiobjective evolutionary algorithms for energy and service level scheduling in a federation of distributed datacenters. *International Transactions in Operational Research*, 24(1-2):199–228.
- Iturriaga, S. and Nesmachnow, S. (2016). Scheduling energy efficient data centers using renewable energy. *Electronics*, 5(4).
- Iturriaga, S., Nesmachnow, S., Tchernykh, A., and Dorronsoro, B. (2016). Multiobjective workflow scheduling in a federation of heterogeneous green-powered data centers. In *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 596–599.
- Iturriaga, S. and Nesmachnow, S. (2015). Multiobjective scheduling of green-powered datacenters considering QoS and budget objectives. In *Innovative Smart Grid Technologies Latin America*, pages 570–573.
- Iturriaga, S., García, S., and Nesmachnow, S. (2014). An empirical study of the robustness of energy-aware schedulers for high performance computing systems under uncertainty. In Hernández, G., Barrios Hernández, C. J., Díaz, G., García Garino, C., Nesmachnow, S., Pérez-Acle, T., Storti, M., and Vázquez, M., editors, *High Performance Computing*, volume 485 of *Communications in Computer and Information Science*, pages 143–157. Springer, Berlin, Heidelberg.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

This chapter introduces the motivation for the work presented in this thesis. After that, it outlines the major contributions and the organization of this thesis.

1.1 State of situation and motivation

Distributed computing systems are key for supporting modern computing demands, currently processing billions of transactions over the internet every day (Khan and Zomaya, 2015). These systems are comprised of many computing resources networked together and comprising a single data center, or several data centers spanning different geographical regions. In the last decade, data centers with thousands of computing resources have been deployed by well-known business organizations, such as Microsoft, Amazon, and Google, among others; and by scientific organizations, such as the National Supercomputing Center in China, the Swiss National Supercomputing Centre in Switzerland, the Oak Ridge National Laboratory and the Lawrence Livermore National Laboratory in United States, and the Joint Center for Advanced High Performance Computing and the RIKEN Advanced Institute for Computational Science in Japan (Chalise et al., 2015).

The energy consumption of data centers has become a major concern for environmental and economic reasons. Van Heddeghem et al. (2014) estimated that the worldwide energy consumption of data centers in 2012 was around 270 TWh with an annual growth of 4.4% between 2007 and 2012. According to Shehabi et al. (2016), data centers in the United States alone consumed

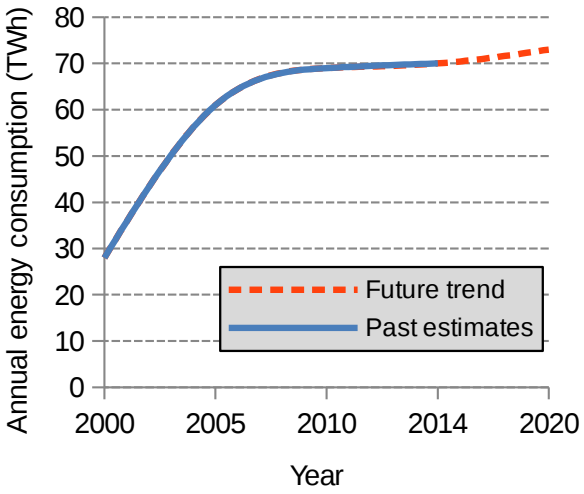


Figure 1.1: Estimated annual energy consumption of data centers in the United States between 2000 and 2020 (Shehabi et al., 2016).

about 61 TWh in 2006, around 1.5% of the total energy consumption of the whole country that year. This is the amount of energy consumed by around 5.8 million average United States households. The energy consumption increased to around 70 TWh in 2014, scaling to around 1.8% of the total energy consumption of the country that year. Figure 1.1 shows the estimated annual energy consumption of data centers in the United States from 2000 to 2020.

Figure 1.1 shows a huge leap in energy consumption from 2000 to 2005, doubling the total energy consumption of data center in just five years. However, since 2008 energy consumption has greatly stabilized in comparison to the first 2000-2005 period. Furthermore, energy consumption since 2008 has stabilized while also meeting a constantly increasing computing demand for data center services. Shehabi et al. (2016) argued this is mainly because of key improvements on energy efficiency and because of the cloud paradigm popularization. On the one hand, the grim energy consumption predictions of the 2000-2005 period motivated hardware manufacturers and data center designers to invest into incorporating energy efficiency techniques in many areas such as server power scaling, storage, networking, cooling systems, etc. This greatly contributed for reducing energy consumption. On the other hand, the increasing popularity of the cloud paradigm motivated the migration of many small in-house data centers to huge outsourced data centers hosting cloud infrastructures.

With the cloud paradigm, efficiency is improved by simply sharing a common computing infrastructure. This paradigm provides a better utilization of the computing resources, reducing their idle time and their energy consumption.

Regarding the future trend, most studies agree energy consumption is expected to continue steadily increasing in the next few years, mainly for coping with the annual network traffic growth expected until 2020 (Cisco Systems, Inc., 2016). However, the accurate growth in energy consumption of data centers is difficult to predict. On the one hand, Shehabi et al. (2016) predicted energy consumption will increase by 4% in the United States on the next few years, reaching 73 TWh in 2020. On the other hand, Andrae and Edler (2015) presented a much grimmer forecast predicting a 4.5% annual increase of the worldwide energy consumption, surpassing 1000 TWh by 2030.

1.2 Research context

Much effort has been put into improving the energy efficiency of data centers. When analyzing the energy consumption of modern data centers, studies have shown that the most power-hungry components in a data center are central processing units (CPUs) of the computing resources and the cooling systems. Several studies show CPUs account for around 30% of the total energy consumption of a data center and cooling systems account for around 35% (Dayarathna et al., 2016; Rong et al., 2016; Shuja et al., 2016a; Song et al., 2015; Zhang et al., 2016). For this reason, many works have addressed the energy efficiency optimization of these two components (Shuja et al., 2016a). An effective well-known approach for optimizing the energy consumption of CPUs consists in intelligently applying dynamic power management (DPM) techniques. These techniques include dynamic voltage and frequency scaling (DVFS) for reducing energy consumption while the CPU is computing, or per-core power-gating and power sleep states for reducing energy consumption while the CPU is partially or totally idle (Pore et al., 2015). As for the cooling system, a number of approaches with diverse scope have been proposed, such as novel cooling techniques (Haywood et al., 2015), optimizing room design (Parnell et al., 2016), optimizing hot-air recirculation (Lin and Deng, 2017), utilizing free cooling (Kim et al., 2017), and recycling waste heat (Zachary Woodruff et al., 2014), among others. Furthermore, in the last few years techniques

for addressing energy efficiency in data centers have been empowered by the widespread dissemination of green energy sources (Deng et al., 2014). Many modern data centers have incorporated on-site or off-site green energy generation into their infrastructures, reducing carbon footprint and operating budget by partially or totally replacing traditional energy (Chalise et al., 2015). However, optimizing the usage of green energy sources is a major challenge itself. Because of their unreliability, data center providers usually require error-prone weather forecasts for predicting green energy availability (Shuja et al., 2016b).

The previously presented techniques contribute for improving the energy efficiency of data centers. However, in order to take full advantage of most of them, it is key to consider an accurate scheduling of the computing workload of the data center (Dayarathna et al., 2016; Shuja et al., 2016b; Zhang et al., 2016). Furthermore, an accurate scheduling algorithm requires a precise modeling of the data center. By modeling characteristics of the data center state (such as temperature, characteristics of computing resources, etc.) and inputs to the data center (such as green energy availability, traditional energy pricing, outside temperature, workload urgency, etc.), the scheduling algorithm may operate the data center most efficiently for a given scenario. However, modeling all these characteristics and the relationship between them is a complex and error-prone matter. On top of the complexity introduced by the modeling, the data center scheduling problem in its general form has been proved to be NP-hard (Rodriguez and Buyya, 2017; Sun et al., 2016), turning the problem of computing an accurate schedule even more difficult when addressing real-world scenarios. The key importance and the complexity of computing an accurate schedule has turned this problem a major challenge to address for modern data center infrastructures.

Figure 1.2 shows the generic schema of a modern data center infrastructure, summarizing some of the previously presented concepts.

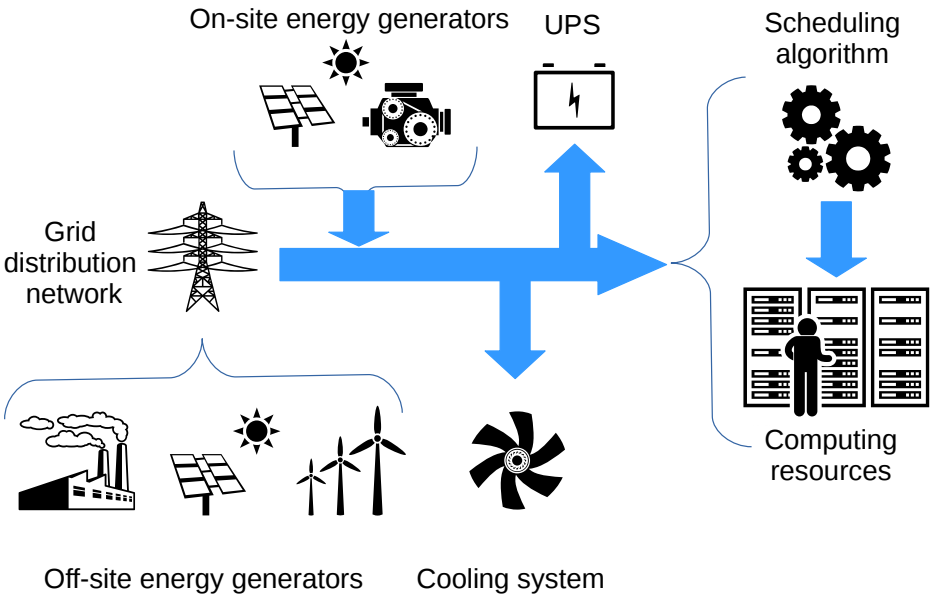


Figure 1.2: Power schema of a modern data center.

1.3 Thesis contribution and organization

This thesis addresses the challenge of scheduling the operation of energy-efficient data centers. It major contributions are the following:

- We present an extensive survey of the current state of the art with an in-depth focus on the most recent works.
- We propose a model for the simultaneous optimization of computing resources and cooling systems.
- We consider modern computing systems, modeling multicore computing resources and renewable energy generators.
- We address several scheduling problems, considering scenarios with a single data center and scenarios with several geographically distributed data centers.
- We propose mathematical formulations for all the proposed problems, simultaneously considering energy efficiency and quality of service (QoS).
- We design a data center model taking into account green energy forecasting, heat dissipation of its computing resources, and effectiveness of its cooling devices.
- We approach the proposed scheduling problems using a multi-objective methodology, computing and studying a Pareto front of schedules that sample the trade-off between the considered objectives.

- We design a set of accurate multi-objective scheduling algorithms for addressing the proposed problems.
- We construct a set realistic and diverse problem instances for evaluating the proposed scheduling algorithms.
- Through extensive statistical analysis, we show the proposed algorithms compute accurate schedules, adequately sampling trade-off solutions.
- We study the impact of execution time and energy consumption uncertainties in the accuracy of different scheduling algorithms.

These contributions are organized in Chapters 2 to 6. Chapter 2 presents a literature review analyzing the current state of the art.

Chapter 3 is based on article Iturriaga and Nesmachnow (2015) and article Iturriaga and Nesmachnow (2016). In this chapter the scheduling problem of controlling power consumption in a single data center is addressed, considering both traditional and renewable energy sources. The proposed scheduling problem consists in simultaneously scheduling the power state of the servers and the cooling system of the data center. The problem takes into account a desired reference power consumption profile, the overall electricity budget, and the QoS provided to its users. Two multi-objective evolutionary algorithms are designed for solving the problem, both hybridized with a greedy scheduling algorithm and a simulated annealing algorithm. Results show the proposed algorithms are able to compute high QoS values and low power profile deviation with average budget reductions ranging from 33% up to 83% when compared to a business as usual scenario.

Chapter 4 is based on the articles Iturriaga et al. (2016) and Iturriaga et al. (2017). This chapter addresses the multi-objective problem of scheduling a large number of workflows in a federation of several geographically-distributed data centers, with each data center being comprised of homogeneous computing resources. The proposed problem considers the simultaneous minimization of three objectives: makespan, energy consumption, and number of workflows violating a service level agreement (SLA). A two-level hierarchical scheduling algorithm is designed for solving this problem. The hierarchical algorithm is comprised of a high-level and a low-level algorithm. The high-level algorithm schedules workflows to different data centers and the low-level algorithm schedules each workflow to the computing resources of its assigned data center. A constraint programming model it is also designed for finding lower bounds for

each objective for medium-sized instances, considering the relaxation of some of the problem constraints.

Furthermore, five heuristics are introduced to address the online version of the problem, and two multi-objective evolutionary algorithms (MOEAs) to address the offline version of the problem.

Chapter 5 presents unpublished work, extending the approach presented in Chapter 4 by considering data centers comprised of heterogeneous computing resources and by considering networking communication. This new formulation provides a more realistic modeling for nowadays data centers. It also extends the experimental analysis by considering a total of 56 heuristic algorithms for online scheduling and 3 MOEAs for offline scheduling. The accuracy of all these newly proposed high-level schedulers is studied considering a set of 100 diverse and realistic problem instances.

Chapter 6 is based on the article Iturriaga et al. (2014). In this chapter a formulation for the energy-aware scheduling problem considering uncertainties is presented. This formulation considers uncertainties in the execution time of tasks and in the energy consumption of the computing infrastructure. Real-world computing workloads are analyzed and workload generation model considering uncertainties is proposed. Furthermore, empirical evaluations are conducted to validate the proposed energy consumption model, and a set of scheduling algorithms considering different strategies are evaluated. Results show uncertainty values in real-world scenarios may significantly affect the accuracy of the scheduling algorithm depending of its scheduling strategy.

Finally, Chapter 7 presents the overall conclusions of the work presented in this thesis and outlines the main lines of future work.

Chapter 2

Towards energy efficiency in data centers: a literature review

In the last decade, the problem of achieving energy efficiency in data centers has become a relevant topic for the scientific community (Chalise et al., 2015). This chapter presents and characterizes the most recent and relevant approaches for improving energy efficiency in data centers.

The next section introduces an overview of the different modeling approaches for addressing the energy efficiency problem in data centers. Section 2.2 presents works dealing with efficiency in computing components. Next, Section 2.3 presents works which consider servers and cooling devices in their model. After that, Section 2.4 presents works which consider renewable energy sources for powering the data center. Finally, Section 2.5 summarizes all the presented works.

2.1 Modeling approaches

Approaches for improving energy efficiency can be classified according to whether they model energy efficiency by considering servers individually, group of servers, whole data centers, or even federations of data centers. Arguably the simplest model considers energy efficiency at server level. However, because of its simplicity this level of modeling fails to properly consider the impact of non-local aspects such as networking infrastructure and heat dissipation, among others. A natural extension for this model consists in considering groups of servers such as blades, racks, or whole aisles of servers.

Modeling a group of servers solves some of the shortcomings of the server-level modeling by considering the interaction between a reduced number of servers. However, it does not model the interaction between servers in different groups. Modeling this interaction is critical for accurately considering characteristics such as heat dissipation and air recirculation. As such, many state-of-the-art approaches consider modeling at data-center level, describing the interactions of all energy consuming devices inside the data center (Dayarathna et al., 2016). Recently, with the popularization of smart grid technologies, much effort has been put to further extend the modeling scope to consider federations of geographically-distributed data centers. This level of modeling allows taking into account smart grid characteristics such as location-dependent pricing schemes and renewable energy availability, among others (Erol-Kantarci and Mouftah, 2015).

The most straightforward approach for achieving energy efficiency consists in optimizing the energy consumption of computing components. Pore et al. (2015) and Dayarathna et al. (2016) provided comprehensive surveys on this topic by reviewing techniques for reducing the energy consumption of the most relevant components at the server level, such as processor, memory, disk, and network interface. Furthermore, they presented various approaches for modeling energy consumption at group-of-servers and data-center levels by considering cooling and networking efficiency. Most of these techniques rely on dynamic power management (DPM) features supported by the hardware such as dynamic voltage frequency scaling (DVFS) (Weiser et al., 1996), low-power states (Meisner et al., 2009), per-core power gating (Leverich et al., 2010), or even turning off servers (Lin et al., 2013). Rong et al. (2016) complemented the works by Pore et al. (2015) and Dayarathna et al. (2016) by thoroughly reviewing several data center efficiency metrics and discussing static and dynamic software techniques for energy efficiency, specifically targeting high performance computing systems. Static techniques include compiler optimization and coding ways, while dynamic techniques deal with resource scheduling, workload management, and networking protocol optimization. Sun et al. (2016) further dived into dynamic techniques and surveyed approaches for workload prediction and allocation, studies global and local resource scheduling methods, and characterizes workloads considering different types of applications. Furthermore, Sun et al. (2016) studied the impact of resource and workload heterogeneity on scheduling strategies.

Finally, Shuja et al. (2016a) considered a different approach and addressed the problem from the perspective of data center design by analyzing energy-efficient design alternatives for the computing, storage, and communication system.

On top of reducing energy consumption of computing components, several works deal with optimizing energy consumption of cooling devices. This strategy is most effective when considering a data-center level modeling. Fulpagare and Bhargav (2015) and Oró et al. (2015a) presented a survey of state-of-the-art designs for air-cooled data centers using either Computer Room Air Conditioning (CRAC) or Computer Room Air Handle (CRAH). They also reviewed state-of-the-art techniques for liquid-cooled data centers. The study analyzes the design of data centers from a thermal perspective and present opportunities for increasing energy efficiency, such as increasing the recommended operational temperature of the data center and evaluating promising cooling strategies. Zhang et al. (2014) and Oró et al. (2015a) reviewed strategies for airside and waterside free-cooling systems, while Oró et al. (2015b) studied the efficacy of airside free-cooling systems in relation to the geographical location of the data center. Ebrahimi et al. (2014) studied the opportunities for waste heat recycling in liquid-cooled data centers. This approach proposes applying the heat dissipated by a data center to different uses, such as heating a neighborhood or a greenhouse, water desalination or purification, and processing biomass fuels, among others. Furthermore, many resource scheduling techniques have been studied for optimizing thermal management. Zhang et al. (2016) and Chaudhry et al. (2015) reviewed many thermal-aware schedulers at group-of-servers and data-center levels. These thermal-aware schedulers aim at minimizing the impact of the dissipated heat by preventing heat concentration in small areas (known as *hotspots*) in order to improve the efficacy of the cooling mechanism.

Recently, powering data centers using renewable energy has become a key approach when addressing energy efficiency problems. Shuja et al. (2016b) presented several case studies of data centers powered by renewable energy, such as Parasol (Goiri et al., 2015a), and introduced a number of open issues and challenges regarding integrating renewable energy to data centers. In this regard, Oró et al. (2015a) reviewed on-site and off-site integration approaches of renewable energy generators and discuss their strengths and weaknesses.

One of the main challenges when considering renewable energy is the uncertainty and the geographical dependency of energy availability. The are two approaches for dealing with the uncertainty of energy availability, the reactive and the proactive approach (Abbasi et al., 2014). A reactive system is totally oblivious of the energy uncertainty and makes use of renewable energy whenever is available and as much as possible. On the contrary, a proactive system (also known as predictive system) relies on forecasting techniques for predicting the amount of renewable energy available in the near future. A well-known strategy for dealing with the geographical dependency of renewable energy is called "following the renewables" (Shuja et al., 2016b). This strategy takes advantage of the global networking availability and considers a federation of geographically distributed data centers. In this scenario the computing workload is simply scheduled to data centers where renewable energy is available at each moment. The work by Rahman et al. (2014) surveyed several resource-scheduling strategies following the geographically distributed approach.

This thesis focuses specifically on improving energy efficiency by accurately scheduling the data center's operation. Several works consider this approach and a number of scheduling strategies have been proposed considering different characteristics of the problem. Lopes and Menasce (2016) provided an extensive taxonomy of scheduling strategies considering the most cited papers in the related literature. Complementing the work by Lopes and Menasce (2016), Wu et al. (2015) and Rodriguez and Buyya (2017) presented precise taxonomies by narrowing the scope and considering just strategies for scheduling precedence-constrained jobs (known as *workflows*). Similarly, Guzek et al. (2015) presented a comprehensive survey of evolutionary-based metaheuristics for scheduling cloud infrastructures.

2.2 Efficiency of computing elements

Barroso and Hölzle (2007) showed that servers in data centers are mostly idle or utilized for up to 50%, hence switching these idle servers to low-power states is key when optimizing energy efficiency. Wang et al. (2011) addressed this issue by proposing PowerSleep, a server-level online heuristic that applies several DPM techniques to react to the computing load and efficiently switch a server to a low-power sleep state.

Experimental results show PowerSleep outperforms DVFS by up to 27%. On top of that, PowerSleep reduces power by up to 40% when compared to PowerNap, a similar heuristic proposed by Meisner et al. (2009).

The low-power sleep state proposed by Wang et al. requires the whole server to be idle. However, this is not always possible. Most of the time, servers are partially used but not completely idle. For addressing these scenarios, Leverich et al. (2010) introduced a DPM technique called *core gating* that reduces the energy consumption of unused cores to almost zero. Experimental results show average energy savings of 20% with no impact on the performance of the system.

Many approaches rely on DPM techniques for reducing energy consumption. Von Kistowski et al. (2015) studied the problem of distributing the computing workload for minimizing energy consumption in single servers and group of servers with multiprocessor architectures. The work states that balancing processor workload is a non-trivial problem and argues that current load balancing approaches are based on generalized assumptions which are not true for several hardware and workload scenarios. The authors proposed an online hierarchical workload algorithm for distributing or consolidating workload in processors and rely on automatic DPM features supported by the hardware for reducing energy consumption. Experimental results show the proposed strategy may reduce energy consumption by up to 10% when comparing with a business as usual scenario.

Although experimental results show the efficacy of the approaches proposed by Wang et al. and Von Kistowski et al., most scheduling research focus on modeling energy efficiency at a data-center level. Such is the case of Mei et al. (2013) who addressed the problem of minimizing makespan and energy consumption in a heterogeneous data center. The scheduling problem consists in scheduling a single job comprised of a set of precedence-constrained heterogeneous tasks. Their work proposes a two-state energy consumption function considering servers to be active or inactive, with servers consuming a fixed amount of energy in either state, disregarding their workload. This is a reasonable assumption in high performance computing (HPC) environments, where tasks are CPU-bound. In such environments, a processor is used at its maximum capacity or not used at all. However, this is not a realistic assumption in other non-CPU-bound environments, not even in HPC environments with multicore processors that can be partially loaded. The work proposes

a greedy heuristic following an offline scheduling approach and compares its efficacy with Bansal et al. (2005) and Hagras and Janecek (2005). Experimental results show the algorithm proposed by Mei et al. (2013) reduces energy consumption by up to 15%.

The work by Guo and Fang (2013) addressed the problem of scheduling a set of independent tasks for minimizing energy budget constrained by average response time in a federation of data centers. These data centers are comprised of homogeneous processors with an on-site energy storage system, such as batteries. The problem considers a heterogeneous scheme for energy pricing by considering data centers to be geographically distributed in different areas with different pricing schemes. The work proposes optimizing the charging of batteries when grid energy is cheap and taking advantage of the stored energy when grid energy is most expensive. Energy consumption is modeled with a linear function where each processor consumes energy proportionally to its load. This energy model differs from the proposed by Mei et al. (2013), providing a more flexible model that is suitable for a wider scope of computing environments. The authors applied the Lyapunov optimization technique (Neely, 2010) for designing an efficient online scheduling heuristic. Experimental results show that the heuristic proposed by Guo and Fang (2013) is effective, outperforming the state-of-the-art heuristics proposed by Qureshi et al. (2009) and Rao et al. (2012a), and reducing energy budget by up to 22% when compared to a business as usual scenario. Although experimental results are promising, the proposed approach is profitable only in environments with highly dynamic pricing schemes where energy cost varies significantly in the scope of a few minutes. This is because traditional energy storage systems can power a data center for a very limited number of minutes before running out of energy.

Following an approach similar to the by Guo and Fang, Shi et al. (2017) addressed the problem of minimizing the completion time of a set of independent tasks constrained by the total energy consumption in a heterogeneous data center. The work proposes a linear energy consumption model where each processor consumes an amount of energy relative to its computing load. Two scheduling heuristics are designed for solving the proposed problem, one aiming at scheduling accuracy and the other at computing efficiency. Both were designed using linear programming relaxation techniques and improved by a local search. Experimental analysis is performed considering problem

instances of up to 200 tasks and up to 500 processors. Results show the proposed algorithms reduce the scheduling completion time by at least 40% when compared with the well-known Min-Min heuristic (Ibarra and Kim, 1977) and two metaheuristic approaches: a genetic algorithm proposed by Kumar and Verma (2012) and a genetic algorithm hybridized with a simulated annealing proposed by Guo-ning et al. (2010). Furthermore, the proposed heuristic oriented for computing efficiency is up to 10 times faster than the oriented on scheduling accuracy, with a 5% loss in scheduling accuracy.

More recently, Jena (2015) considered the multi-objective problem of minimizing makespan and energy consumption in a federation of heterogeneous data centers when scheduling a set of jobs comprised of many precedence-constrained tasks. Regarding the power consumption model, Jena considered a simple two-state linear model where processors may be active or inactive. An active processor consumes a fixed amount of energy, while an inactive processor does not consume energy at all (i.e. it is powered off). Jena (2015) proposed a batch scheduling solution based on the multi-objective particle swarm optimization algorithm (Coello Coello and Lechuga, 2002). Experimental analysis considers problem instances of up to 400 processors and 360 tasks and is performed using CloudSim, a cloud simulator proposed by Calheiros et al. (2011). Results show the algorithm proposed by Jena reduces energy consumption by 30% and makespan by 25% in average, when compared with a greedy heuristic. Although the work proposes a multi-objective approach, the authors fail to thoroughly study the trade-off between energy consumption and system performance using a Pareto-based approach. The work reports experimental results on a per-objective basis, and does not study how these objectives relate to each other.

Kaushik and Vidyarthi (2016) solved the multi-objective scheduling of a set of jobs, each comprised of precedence-constrained tasks, for optimizing reliability, energy consumption, and load balancing in a federation of data centers with homogeneous processors. Energy consumption is modeled with a linear function considering the energy consumption of active network links and modeling the energy consumption of each processor by two states: active or asleep. Processors are assumed to consume a fixed amount of energy in each state, similar to the approach followed by Mei et al. (2013). Kaushik and Vidyarthi considered the migration of jobs among data centers to address the load balancing optimization objective. The work considers an evolutionary

approach and proposes a multi-objective offline scheduler based on NSGA-II (Deb et al., 2002) and a single-objective batch scheduler based on a simple genetic algorithm. Experimental analysis is performed considering instances of up to 5 data centers with up to 100 processors each, and up to 3 jobs with up to 25 tasks each. Results show that the proposed schedulers are accurate and compute schedules with different characteristics. On the one hand, the simple genetic algorithm outperforms the scheduler based on NSGA-II in a single-objective basis. On the other hand, NSGA-II computes a set of trade-off solutions that the genetic algorithm is unable to compute. Hence, the proposed methods complement one another in exploitation and exploration. If the scheduling goal is strongly biased towards one of the objectives, then the genetic algorithm is the most accurate method. However, if the desired schedule should balance both objectives, then NSGA-II is most adequate.

When dealing with precedence-constrained jobs, a well-known energy-optimization approach consists in taking advantage of DVFS by applying a slack reclamation technique. This strategy heavily relies in the nonlinear energy consumption model of the complementary metal-oxide semiconductor (CMOS) (Weiser et al., 1996; Ishihara and Yasuura, 1998). This model shows that a linear reduction in performance translates into a quadratic reduction in energy consumption of the processor. Hence, methods following this approach usually consider deadlines and aim at reducing the performance of the system, delaying the finishing time of tasks as much as possible without violating their deadlines. Such is the case of the approaches recently proposed by Garg and Singh (2016), Tang et al. (2016), Xie et al. (2016), Chen et al. (2016a), and Sajid and Raza (2017).

Garg and Singh (2016) addressed the problem of minimizing energy consumption when scheduling a single job comprised of precedence- and deadline-constrained tasks in a DVFS-enabled heterogeneous data center. Garg and Singh proposed an energy consumption model considering processor and network energy consumption. Network energy consumption follows a linear model, while the processor energy consumption follows a nonlinear function based on the CMOS power consumption model. The authors proposed computing the schedule in three phases, applying three offline greedy scheduling algorithms sequentially. Experimental results show the proposed solution reduces energy consumption up to 45% with just a 5% increase in execution time when compared to previous heuristic approaches proposed by Lee and Zomaya (2009),

Baskiyar and Abdel-Kader (2010), and Zong et al. (2011). Garg and Singh argue the proposed algorithm computes an accurate trade-off schedule. However, it is difficult to measure the accuracy of the proposed algorithm without a proper multi-objective study showing how the objectives relate to one another.

Tang et al. (2016) dealt with the problem of minimizing the energy consumption of a single job comprised of precedence-constrained and communication-aware tasks in a heterogeneous DVFS-enabled data center. The authors addressed a problem similar to the one addressed by Garg and Singh (2016). Also similar to Garg and Singh, Tang et al. considered a nonlinear energy consumption model based on processor energy consumption and considered the time required for data communication among related tasks. Furthermore, Tang et al. defined a deadline constrain on the job, i.e. a makespan constraint on its tasks, following a slack reclamation approach. The work proposes a simple and efficient offline DVFS-aware heuristic based on HEFT (Topcuoglu et al., 2002). Experimental results are compared with two state-of-the-art scheduling heuristics: HEFT and EES (Huang et al., 2012). Several problem instances comprised by a number of tasks ranging from 20 up to 400 and a number of processors from 2 up to 48 were created for the experimental evaluation. Results show the algorithm proposed by Tang et al. reduces energy consumption by 7% in average and up to 10% in the best case scenario when compared to EES. However, the proposed scheduler trades this reduction in energy consumption with an average 2% increase in execution time when compared with HEFT. This is an expected compromise since the slack reclamation approach considered by Tang et al. increases energy savings by reducing system performance as much as possible, subject to the deadline constraint. Overall, the scheduler provides an adequate reduction in energy consumption while meeting the job deadline.

Also in the same line, Xie et al. (2016) addressed the problem of minimizing the makespan of a single job comprised of precedence-constrained communication-aware tasks in a DVFS-enabled heterogeneous data center subject to the total energy consumption. This is quite similar to the approach proposed by Tang et al., but considering makespan as an objective and energy consumption as a restriction instead the other way around. This work follows a nonlinear energy consumption model based on processor energy consumption, further considering a processor sleep state for reducing energy consumption. Xie et al. proposed a offline HEFT-based heuristic and compared it with

HEFT and ECS. Experimental evaluation is performed considering 10 different problem instances comprised by a number of tasks ranging from 39 up to 1274. Results show the proposed heuristic outperforms HEFT and ECS reducing energy consumption in every problem instance by 32% in average and reducing makespan in large-sized instances by up to 14%.

The approaches proposed by Garg and Singh, Tang et al., and Xie et al. are accurate and computationally efficient methods, however these approaches fail to study the relation between energy consumption and execution time. Only one of these characteristics is optimized at a time, while the other is modeled as a fixed constraint. Taking this into consideration, the articles by Chen et al. (2016a) and Sajid and Raza (2017) addressed the simultaneous optimization of makespan and energy consumption.

Chen et al. (2016a) proposed to minimize makespan and energy consumption of a single job comprised of precedence-constrained and communication-aware tasks in a DVFS-enabled data center. Chen et al. formulated two separate single-objective optimization problems: one for minimizing makespan subject to total energy consumption of the data center, and the other for minimizing energy consumption subject to makespan of the schedule. The proposed model considers only the processor for computing the energy consumption of the system. Like in the approach proposed by Garg and Singh (2016), processor energy consumption is modeled using a nonlinear function based on the CMOS power consumption model. Chen et al. proposed an offline hyper-heuristic algorithm comprised of seven low-level heuristics for searching feasible scheduling solutions and a high-level quantum-based heuristic for guiding the search. The proposed hyper-heuristic is compared with several state-of-the-art schedulers such as HEFT, ECS proposed by Lee and Zomaya (2011), the genetic algorithm proposed by Bozdag et al. (2009), and the tabu-search hyper-heuristic proposed by Burke et al. (2003). Experimental evaluation is performed considering several jobs comprised by a number of tasks ranging from 16 up to 256. Results show the proposed hyper-heuristic outperforms HEFT and ECS improving makespan by up to 19% and energy consumption by 11%. On top of that, it is able to outperform the genetic algorithm and the tabu-search methods in 15 of the 36 problem instances, while it is not outperformed in any of the remaining 21 instances.

Sajid and Raza (2017) considered the problem of scheduling a batch of stochastic jobs in a DVFS-enabled heterogeneous data center using a multi-

objective approach. Each job is comprised of precedence-constrained tasks whose execution times are not precisely known beforehand. Instead, the execution time of a task is known only after it finishes its execution. Only the probability distribution of its execution time is known beforehand. The goal of this work is computing robust schedules, simultaneously minimizing the expectation and variance of both makespan and energy consumption. Energy consumption is computed considering just the processor consumption and following a nonlinear energy consumption model based on the CMOS power consumption model. The authors proposed an online batch-oriented multi-objective evolutionary algorithm based on the hypervolume estimation algorithm for multi-objective optimization (HypE) proposed by Bader and Zitzler (2011). Experimental evaluation is performed by comparing the computed results of the proposed HypE-based algorithm with an algorithm based on NSGA-II, and the heuristics SHEFT (Tang et al., 2011), HEFT, and ECS. Results show the HypE-based method outperforms all other methods by up to 11% in terms of average turnaround time and average energy consumption.

Although all these approaches are based on slack reclamation techniques, only the approach proposed by Sajid and Raza deals with the scenario of a general purpose data center were a set of jobs must be scheduled. The remaining approaches proposed by Garg and Singh, Tang et al., Xie et al., and Chen et al. deal with the scheduling of a single job that has the entire data center at its disposal.

2.3 Cooling efficiency

Computing servers and cooling devices are the most energy consuming components in modern data centers (Chaudhry et al., 2015). Hence, just addressing the efficiency of computing elements is not enough for a true energy-efficient data center. Wang et al. (2010) studied the importance of a unified management approach by taking into account computing and cooling parameters. Wang et al. argued thermal models are key for improving the efficiency of data centers and must provide feedback to the scheduling algorithm. The work by Wang et al. studies the behavior of a single CRAC-cooled 16-server blade enclosure with the objective of minimizing the energy consumption subject to operational budget. A dynamic thermal model for the blade enclosure is presented in this work, taking into account server disposition and cooling

efficiency by applying heat transfer theory for thermal resistance and energy balance via a lumped capacitance method. On top of that, energy consumption is modeled using a linear function considering the load of each processor. Wang et al. showed the effectiveness of cooling devices is highly dependent of the physical layout of the data center, since cooling is more effective for processors closest to the cooling devices. This should be taken into consideration, even when dealing with data centers with homogeneous processors. Even more, highly loaded processors that are physically close generate hot spots of dissipated heat that may be much harder to cool down because of the air circulation flow. The authors propose to tackle these issues with an online scheduling algorithm based on simulated annealing (Kirkpatrick et al., 1983) for efficiently solving the workload placement problem. Experimental results show the proposed strategy reduces cooling energy consumption by 42% when compared with a non-thermal-aware scheduler. Furthermore, this work proposes a hierarchical control architecture to extend its solution for controlling multiple blade enclosures and even a whole data center.

Several works extend the rack-level modeling proposed by Wang et al. in order to deal with whole data centers. The general optimization strategy consists in maximizing cooling efficiency using workload placing techniques, and reducing cooling usage as much as possible subject to a maximum operating temperature. A widely accepted thermal operating guideline is published and regularly updated by the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE Technical Committee, 2011).

Lee et al. (2017) tackled the problem of scheduling independent tasks for minimizing energy consumption in a CRAC-cooled heterogeneous data center subject to a maximum operating temperature and considering task migration. The authors followed the heat dissipation model proposed by Chandra et al. (2002) and the heat extraction model by Moore et al. (2005). A three-state energy consumption model is considered taking into account processors in active state, inactive state, and off state. Processors consume the maximum supply of energy when active, the minimum supply when inactive, and no power when off. Lee et al. proposed a greedy online heuristic and compare its performance against five baseline heuristics, and two state-of-the-art heuristics proposed by Moore et al. (2005) and Lee et al. (2012). Results show the proposed algorithm is the most effective method, reducing energy consumption by 12% compared with Moore et al. (2005) and by 3% compared with Lee et al. (2012).

The work does not consider quality of service as a problem objective. However, quality of service is clearly a conflicting objective because of migrating tasks and switching off servers. This could impact negatively on the accuracy of proposed method.

Al-Qawasmeh et al. (2015) addressed two scheduling problems: maximizing quality of service subject to energy consumption, and minimizing energy consumption subject to quality of service. The problem model considers a set of deadline-constrained independent tasks, and a heterogeneous CRAC-cooled data center. The quality of service is defined as the number of tasks' deadlines met. Also, both problems are subject to maximum temperature operating constraint of the data center. The considered thermal model is based on the one proposed by Tang et al. (2006), a low-complexity linear model which uses a cross-interference coefficient matrix to model the heat transfer ratio among servers. The authors presented a multi-core computing model where energy consumption is modeled using a nonlinear function based on dynamic power states for each core of the data center. Two online heuristics are proposed–one for each problem–, both based on a relaxed formulation using mixed-integer nonlinear programming. Results show these techniques achieve a 17% improvement in quality of service and around 9% improvement in power consumption when compared to a greedy scheduling heuristic.

Lin and Deng (2017) solved the problem of minimizing the power consumption of the cooling system in a homogeneous data center when executing independent tasks. The problem considers a data center comprised of blade servers organized in racks and cooled by a CRAC system. Lin and Deng proposed a thermal model considering heat recirculation model for optimizing cooling efficiency and avoiding hot spots. Two task-scheduling algorithms are introduced: an enhanced genetic algorithm and an online greedy heuristic. The genetic algorithm proposes to enhance the performance of the genetic algorithm by not performing any crossover operation and just performing mutations. On top of that, initial population is initialized randomly when it could be easily initialized using the proposed greedy heuristic. These poor decisions turn the evolutionary process close to a random walk. Experimental analysis shows that both proposed scheduling algorithms reduce energy consumption up to 60% when compared to a random walk scheduling algorithm. However, the greedy heuristic outperforms the enhanced genetic algorithm, reducing the energy consumption between 2% and 5%. This is an expected result given the

modifications performed to the behavior of the genetic algorithm.

The work by Manousakis et al. (2015) significantly differs from the previously presented works because it considers a free cooling system that takes advantage of outside air for cooling the data center. The work addresses the scheduling problem of minimizing cooling cost in homogeneous DVFS-enabled data centers considering CRAC and free cooling, subject to quality of service. Cooling cost refers to the CRAC system and is comprised of purchasing cost, operating cost, and cost of replacing failed hardware. Tasks are considered to be independent and are scheduled online. Free cooling is provided to the data center leveraging outside air when outside temperature is adequate. Manousakis et al. argue traditional data centers commonly provision CRAC systems for dealing with the highest peak of demand. However, these peaks are often rare, Manousakis et al. say. Hence, they proposed to under-provision the cooling system and make use of free cooling to cope with this deficit whenever possible. Because of this strategy, the cooling system will not be able to satisfy cooling demands during periods of peak demand. When cooling demands are not satisfied, the authors propose to either reduce the performance of the system or let the temperature to increase considering an increase in IT equipment failure. This approach makes use of an expected usage power profile and outside conditions for dimensioning the CRAC system. Manousakis et al. (2015) proposed the use of DVFS, workload deferral, and migration techniques, and defines the quality of service as the number of times the system applies these techniques. The energy consumption model considers two states: active and inactive. When a processor is in active state, energy consumption is a linear function relative to the processor load. Otherwise, when in inactive state, processors are in sleep state and are considered to consume near-zero energy. An online approach based on sequential quadratic programming is proposed for solving this problem. The experimental evaluation is performed in the Parasol data center (Goiri et al., 2015b) using an auto-regressive thermal model with exogenous input. Results show the proposed scheduling algorithm is able to reduce cooling costs by up to 55% when compared to business as usual scenario. Although the budget reduction results are promising, this study fails to consider the possible reduction in the fidelity of its users when tasks are affected by reduction in performance or hardware failures.

A number of works focus on studying the relation between cooling efficiency and its impact on the computing elements, which may be measured from

a user perspective with a quality-of-service related metric, or from a system perspective with a resource utilization metric. Meng et al. (2015) addressed the scheduling problem of minimizing communication cost and cooling energy consumption of parallel jobs comprised of communication-intensive tasks in a CRAC-cooled homogeneous data center. The problem formulation defines the communication cost as the average network distance between all pair of tasks in a job. Also, all tasks in a parallel job must start simultaneously and execute in parallel. Jobs follow an all-to-all homogeneous communication pattern where each task communicates with all the other tasks in the job, such as in the Fast-Fourier-Transform application (Kumar et al., 2008). The considered energy consumption model follows a three-state model: computation-intensive state, communication-intensive state, or idle; each state consuming a fixed supply of energy. Finally, Meng et al. followed the thermal model proposed by Tang et al. (2006). The scheduling solution proposed by Meng et al. (2015) combines the online algorithms previously proposed by Tang et al. (2008) for temperature minimization and by Bender et al. (2008) for communication cost minimization. Experimental results show the scheduler proposed by Meng et al. reduces cooling power by up to 42% with a communication cost gap of 3% when compared with Tang et al. and Bender et al.

Habibi Khalaj et al. (2015) proposed the scheduling problem of minimizing hot spots and maximizing processor utilization in a CRAC-cooled homogeneous data center subject to operating thermal constraints. The work studies the relation between both objectives, aiming for a set of accurate trade-off schedules. Habibi Khalaj et al. proposed an accurate thermal model based on a reduced order model of the Navier-Stokes equations, considering buoyancy and recirculation around racks. Furthermore, this lightweight model is verified using a precise computational fluid dynamic model. The energy consumption model follows a linear model where energy consumption increases linearly with the processor load. Habibi Khalaj et al. did not model the workload to be comprised as tasks as individual units, but as a stream of processing load. This is a reasonable simplification when dealing many short-lived tasks, such as the ones processed by web servers. A multi-objective particle swarm optimization (Coello Coello and Lechuga, 2002) for offline scheduling is proposed. Results show the proposed method increases CRAC performance by 17% and processor utilization by 10% when compared with a business as usual scenario.

Goudarzi and Pedram (2016) addressed the problem of scheduling independent tasks for minimizing the budget of the cooling system and maximizing the fulfillment of the service level agreement, defined by the maximum response time. Tasks are to be scheduled in a data center consisting of a set of thermal-isolated containers, each one comprised of homogeneous servers and cooled by an independent CRAC system. Goudarzi and Pedram described the energy consumption with a linear function of the server load and considers the thermal model proposed by Tang et al. (2007). Finally, Goudarzi and Pedram proposed a hierarchical online scheduling approach consisting of a set of greedy heuristics improved by a local search. Experimental results show the proposed approach reduces the budget of the cooling system by 43% in average when compared to a scheduling algorithm based on the work by Verma et al. (2008).

Considering a cooling model introduces a geographic location challenge. This challenge arises because the efficiency of cooling systems depend on outside weather conditions, mainly on temperature and humidity (Oró et al., 2015b; Song et al., 2015). A number of articles address this challenge, in particular the ones by Polverini et al. (2014), Rajabi et al. (2014), and Xu et al. (2015). These articles study the scheduling of tasks in a federation of geographically-distributed data centers.

Polverini et al. (2014) proposed to minimize energy budget and maximize fairness when scheduling a set of independent jobs in a federation of heterogeneous data centers cooled by a CRAC system. The authors considered each task to be owned by a given user and each user to have a certain weighted importance in the system. The fairness function is defined for each moment in time and is maximized when every user has a number of computing resources assigned to its executing tasks according to its relative importance in the system. The energy consumption of the cooling system varies among data centers due to the efficiency of the CRAC system. For energy consumption modeling, this work proposes a linear model relative to the processor load, and for thermal modeling it applies the model proposed by Mukherjee et al. (2009). Polverini et al. proposed an online algorithm based on the Lyapunov optimization technique and compares its experimental results with the algorithm proposed by Ren et al. (2012). Results show the Lyapunov algorithm reduces budget by 10% in average, mainly due to reduction in cooling energy consumption.

Rajabi et al. (2014) addressed the problem of scheduling independent deadline-constrained tasks for minimizing CO_2 emission and maximizing the profit in a federation of heterogeneous DVFS-enabled data centers considering a CRAC system. The energy consumed by the data center is modeled as the energy consumed by its processors using a nonlinear CMOS power model. This work proposes a thermal model based on a linear simplification of the model proposed by Moore et al. (2005). Rajabi et al. proposed an online hierarchical approach comprised of a high-level metaheuristic and a low-level greedy heuristic. Experimental results show that the proposed method increases profit by 9% and reduces CO_2 emission by 17% in average, when compared with the multi-objective genetic algorithm proposed by Kessaci et al. (2013). The authors focus on studying results one objective at a time and do not present a Pareto-based comparison.

Finally, Xu et al. (2015) addressed the multi-objective scheduling problem of minimizing the energy consumption in a federation of heterogeneous data centers, each with homogeneous processors, and cooled by CRAC and free-cooling systems. The problem also considers the quality of service by modeling the user tendency to leave the system as latency increases. Xu et al. argue the efficiency of free-cooling system heavily depends in spatial and temporal conditions. Hence, the formulation takes into account the geographical distribution of the data centers in the federation. The work considers a linear energy consumption model and an empirical regression-based thermal model. Xu et al. proposed an analytic scheduling method based on the alternating direction method of multipliers (Bertsekas and Tsitsiklis, 1989) to compute a static routing scheme every hour. This method is compared with a baseline method similar to the scheduling algorithms proposed by Lin et al. (2013) and Rao et al. (2010), among others. Experimental evaluation shows the proposed scheduler outperforms the baseline method reducing cooling energy consumption by 17% in average.

2.4 Renewable energy sources

Deng et al. (2014) argues that renewable energy sources are increasingly popular for reducing energy-related budget and carbon footprint in data centers. However, efficiently incorporating these energy sources is still a major challenge due to their unreliability. Because of their unreliability, renewable en-

ergy sources are naturally suitable for scenarios where the data center is able to quickly adapt to energy fluctuations. Ghamkhari and Mohsenian-Rad (2013), Abbasi et al. (2014) and Toosi et al. (2017) studied the suitability of renewable energy source for powering data centers that exclusively execute short-lived computing jobs, such as web requests. Most of these works show that a simple reactive approach is sufficient for obtaining significant improvements in energy efficiency.

Ghamkhari and Mohsenian-Rad (2013) considered the problem of scheduling independent homogeneous jobs for minimizing energy-related budget in a homogeneous data center considering two different scenarios, with and without on-site renewable generation. The work considers wind turbines for on-site energy generation. The problem model takes into account the quality of the service in the budget equation by including a revenue factor. Ghamkhari and Mohsenian-Rad proposed a simplified cooling model. Cooling energy consumption is computed based on the power usage effectiveness (PUE) value of the data center, where the PUE value represents the average ratio of energy consumed by the data center that is effectively used for computation. This approach fails to consider thermal characteristics, such as hot spots and air circulation that are directly affected by the load distribution determined by the scheduler. As shown by Wang et al. (2010), these thermal characteristics greatly affect cooling demands that in turn contribute–by definition–to the PUE value. Hence, approximating the energy consumption of the cooling system with the PUE value provides a simple but potentially inaccurate approach. The formulation defines a three-state energy consumption model with active, idle and off states. Energy consumption is fixed for each state, being zero for the off state. The number of switched on and off processors is updated every 15 minutes. Furthermore, Ghamkhari and Mohsenian-Rad did not deal with prediction of renewable energy and proposed a reactive approach. The article considers short-duration computing jobs and applies queuing theory for modeling the problem. It shows this model to be tractable and propose applying convex programming techniques for solving it. Experimental results show the proposed method achieves an average of 4% optimality gap, and is able to outperform the methods proposed by Liu et al. (2011) and Rao et al. (2012a). On top of this, results show budget can be significantly reduced when taking advantage of on-site renewable generation, even with a simple reactive approach.

Abbasi et al. (2014) addressed the problem of minimizing the energy budget by scheduling a set of independent homogeneous jobs in a federation of data centers with homogeneous processors, on-site renewable energy generators, and energy storage systems. Quality of service is modeled as a constraint considering a threshold for the response delay of each job. The authors proposed a simple two-state energy model considering active and off states. Processors in active state consume a fixed amount of energy, while processors in off state consume zero energy. Renewable energy generation is predicted using a seasonal auto regressive integrated moving average model (Abraham and Ledolter, 2008) calibrated using historical data. Abbasi et al. modeled the problem as a flow constrained problem and addressed it using linear programming techniques. Experimental results using real-world data show the proposed online algorithm reduces the energy budget by 40% in average when compared with a uniform load balancer algorithm.

Toosi et al. (2017) proposed the problem of scheduling independent jobs for minimizing grid energy consumption in a federation of data centers with homogeneous processors and considering on-site renewable energy generation. The authors did not deal with energy consumption, thermal, and energy generation forecast models. Instead, they proposed an online reactive solution that actively monitors energy consumption and energy generation for each data center and reacts to changes in the scenario. This is a feasible approach since the problem model considers a workload comprised of many short-lived jobs. Experimental results dealing real-world data show the proposed online solution outperforms the approach proposed by Le et al. (2010), increasing renewable energy consumption by 8% and reducing operational costs by 7% in average.

The presented results show renewable energy is promising for powering data centers executing short-lived computing jobs. However, these results are not applicable to general purpose data centers. Scheduling general purpose data centers usually require a proactive approach where the scheduling is based on renewable energy generation forecasting. Next, we present several works following such approach.

Goiri et al. (2015a) addressed the problem of scheduling precedence- and deadline-constrained jobs for minimizing energy budget in a homogeneous data center with on-site renewable energy generators. The work proposes an empirical energy consumption model based on previous executions of the workload, requiring test runs of the considered jobs. On top of that, this model consid-

ers processors to be active, idle or inactive. Inactive processors are put in a fast-transitioning sleep mode; with transitioning in or out of this mode requiring around 7 seconds. Also, processors in idle state consume around 8.6 W, whereas in active state may consume up to 150 W. Regarding renewable energy generation, Goiri et al. considered a realistic 2-day ahead forecasting model based on historical data for prediction. An efficient online scheduling heuristic is proposed using a rolling-horizon approach, with a 15 minutes time slot. Experimental workloads are constructed synthetically based on real-world jobs with arrivals following a Poisson distribution. Experimental results show the proposed scheduler can reduce energy budget by up to 20% when compared to a variant of the EASY backfilling scheduler proposed by Lifka (1995).

Yu et al. (2015) tackled the problem of scheduling independent jobs for minimizing energy budget in a federation of data centers with homogeneous processors, an energy storage system, and both conventional and renewable energy generators. Quality of service is modeled as a constraint, considering a threshold for the average response delay for the jobs. Yu et al. studied a robust scheduling solution by considering grid power outages. Energy consumption of the processors is modeled by a linear function of the computing load, while the energy consumption of the cooling system is approximated with the PUE value of the data center. Regarding the renewable energy generation, the work considers the forecasting model proposed by Damousis et al. (2004) for approximating the energy generation. Yu et al. proposed a Lyapunov-based online heuristic and study the explicit trade-off between operational cost and battery investment cost. Experimental results show the proposed strategy can reduce energy budget by up to 3% with no renewable energy generation, and by up to 50% when considering renewable energy generation, when compared with the method proposed by Rao et al. (2012b).

Paul et al. (2016) extended the work by Lin et al. (2013) by addressing the problem of scheduling independent homogeneous jobs in a federation of homogeneous data centers with on-site renewable energy generators. The problem proposes the minimization of the weighted linear combination of energy budget, processor switching-on cost, response time, and carbon emissions. The energy consumption of the cooling system is approximated by the PUE of the data center, as proposed by Ghamkhari and Mohsenian-Rad (2013) and Yu et al. (2015). Paul et al. proposed a linear energy consumption model where the processor energy consumption is relative to its load, considering a base idle-state

energy consumption. On top of that, unused processors are turned off for further energy saving. The model considers the switching-on cost representing the energy consumed by the processor when powering on. However, the switching-off cost is not considered in this work for two reasons. First, because Paul et al. argue the switching-on cost is significantly larger than the switching-off cost which in turn is negligible. Second, because the switching-off cost can be incorporated into the switching-on cost simplifying the modeling. For dealing with renewable energy generation a reactive approach was considered, hence no forecasting model was required. Finally, response time was taken into account by over-provisioning the data center processors by 10%. Two formulations for this problem were constructed considering different switching-off cost models. Both formulations are convex optimization problems and optimal solutions can be computed in polynomial time. Paul et al. proposed two online algorithms based on convex optimization techniques and compares them with a baseline uniform load balancer, which Paul et al. argue is still being used by some data centers in practice. Experimental results considering real-world data yield the proposed algorithms reduce energy budget by up to 94% and switching cost reduction by up to 91% when compared the baseline algorithm.

Chen et al. (2016b) addressed the problem of scheduling independent jobs for minimizing the operational cost subject to a delay-related quality of service in a homogeneous data center cooled by chilled water and free cooling. The problem model considers on-site renewable energy generators, conventional diesel generators, and energy storage units. The energy consumption models of the chilled-water and free-cooling systems are based on heat transfer theory and the model proposed by Patel et al. (2006). The energy consumption is modeled using a linear function based on the processor load where idle processors do not consume energy at all. For this to be a realistic assumption processors should be switched off, in which case the time and energy required for the switching must be taken into account, just like the approach proposed by Paul et al. (2016). If processors are not switched off, then their energy consumption and heat dissipation while idle should be considered for the energy storage and the cooling systems models. Regarding the renewable energy generation, the work considers a reactive approach for the renewable energy generation and a smart power grid where energy surplus is sold by injection into the power grid. Applying a problem relaxation approach and a stochastic subgradient solver, Chen et al. designed an online scheduling solution, three

baseline schedulers for comparison, and an offline exact solution. Experimental results dealing with real-world scenarios show the proposed batch scheduler reduces operational cost by 24% in average when compared with the best baseline scheduler, computing an average optimality gap of 10% when compared to the exact offline scheduler.

Later, Chen et al. (2016c) considered the scheduling problem for minimizing energy budget in a federation of homogeneous data centers with on-site energy storage, conventional generator and renewable energy generator. Unlike the previous work of the authors, this work mainly focuses in obtaining robust schedules taking into account the uncertainty of renewable energy sources. The work considers interactive and non-interactive jobs. Interactive jobs must be attended immediately while non-interactive are deferrable but within a certain deadline. Furthermore, non-interactive jobs provide revenue when processed in time. Data centers in the federation are cooled using CRAC and free cooling systems. A nonlinar power consumption model was proposed based on the work by Wierman et al. (2012) and cooling requirements are approximated as a polynomial function of the consumed energy. Regarding the renewable energy generation forecasting, Chen et al. followed the work by Zhang et al. (2013) and considered a polyhedral uncertainty set based on historical measurements for modeling forecasting. The article shows the problem to be non-convex and propose solving a relaxed problem formulation using the dual decomposition method. Experimental results show the proposed online algorithm reduces the worst-case budget by 19% when compared to a non-robust algorithm, providing a schedule significantly more robust to errors of renewable energy forecasting.

Kiani and Ansari (2016) solved the problem of scheduling independent homogeneous jobs for maximizing profit in a federation of homogeneous data centers with on-site renewable energy generators. In this formulation, energy consumed from the power grid reduces the profit of the system. For modeling this situation, Kiani and Ansari considered a linear energy consumption model which is a function of the processor workload. The energy consumption model takes into account the cooling system by incorporating the PUE value as a factor into the model. Renewable energy generation follows a reactive approach, similar to the work by Ghamkhari and Mohsenian-Rad (2013). Kiani and Ansari argues that a simple forecasting method is reasonably accurate when using a small enough time slot. Jobs are grouped in a finite number of service types, each type with a different SLA represented by a maximum waiting time.

Every job processed within its SLA produces revenue, meaning the quality of service of the system is included in the profit equation as revenue. Kiani and Ansari proved this problem to be convex and propose an online strategy for solving it. Experimental results show the proposed strategy increases the profit by up to 20% when compared to Ghamkhari and Mohsenian-Rad (2013).

Anastasopoulos et al. (2016) addressed the problem of scheduling independent jobs for minimizing the nonrenewable energy consumed in a federation of CRAC-cooled homogeneous data centers with on-site renewable energy generators. The model approximates the energy consumption of the data centers considering the energy consumption of three main components: processor, network interface, and cooling system. The work considers the network energy consumption model proposed by Katrinis and Tzanakaki (2011). Regarding the processor energy consumption, Anastasopoulos et al. proposed a linear model based on the processor workload. Finally, cooling energy consumption is taken into account by simply doubling the energy consumption of processors when computing the total energy consumption of the system. This approach is similar to the ones proposed by Yu et al. (2015), Paul et al. (2016), and Kiani and Ansari (2016), in that it considers the PUE metric for approximating the energy consumption of the cooling system. Prediction of renewable energy generation if performed using an auto-regressive moving average model (Wellons et al., 2010) based on the analysis of historical data. A batch scheduling is proposed based on sample average approximation (Kleywegt et al., 2002) and subgradient method. Experimental results show the proposed algorithm is able to reduce nonrenewable energy consumption by 10-60% when compared with four simple baseline algorithms.

Peng et al. (2017) proposed the problem of scheduling independent jobs for minimizing the energy budget in a federation of heterogeneous data centers considering on-site renewable energy generators. The authors proposed a two-state energy consumption model considering on and off states following the approach by Lin et al. (2013). When in on state, processor energy consumption is modeled with a linear function of the load of the processor. When in off state, energy consumption is zero but a state-transitioning energy consumption is considered every time a processor switches state. Renewable energy generation is estimated using an exponentially weighted moving average technique (Hunter, 1986) with historical data. An evolutionary algorithm is proposed for addressing the online scheduling problem, and two greedy heuris-

tics are designed for comparison. Experimental results show this approach reduces energy budget between 8% and 10% when compared with the designed heuristics.

Lei et al. (2015) considered the multi-objective problem of maximizing the utilization of renewable energy, minimizing the energy budget, and maximizing the quality of service in a homogeneous data center with on-site renewable energy generators. The problem deals with independent jobs considering execution priorities and deadlines. Jobs are classified as crucial or non-crucial according to their priority, where crucial jobs have execution priority over non-crucial jobs. Lei et al. considered a two-state model for energy consumption where active and idle processors consume a fixed amount of power. Quality of service is defined considering the relative number of jobs whose deadlines are met. An error-prone short-term renewable energy prediction method was simulated by introducing disturbances into historic solar generation records. An online greedy heuristic was proposed for computing an efficient trade-off solution, and three single objective heuristics were designed for comparison– one for each problem objective–. Experimental results using real-world data show the proposed scheduler provide an accurate trade-off when considering all objectives simultaneously. In average, the solution computed by the proposed algorithm presents a renewable energy utilization rate of 94%, deadline satisfaction of 99%, and the second best operational budget. The proposed algorithm is outperformed only at the operational budget objective by the budget-oriented heuristic at the expense of a 4% renewable utilization rate. Although the problem formulation considered multiple objectives, the proposed greedy scheduling strategy does not explore the trade-off between them, computing a single solution at a time.

Later, Lei et al. (2016) extended their previous work to study the trade-off between renewable energy utilization, quality of service and system utilization considering a multi-objective scheduling strategy. This work addresses the scheduling problem of maximizing renewable energy usage and satisfaction rate, and minimizing makespan and total energy consumption of independent jobs in a heterogeneous federation of DVFS-enabled data centers considering on-site renewable energy generators. Energy consumption is modeled considering just the processor and using a nonlinear function based on the CMOS power consumption model. Like in their previous work, renewable energy prediction is simulated introducing disturbances into historic solar generation

records. The most notable difference with their previous work is the proposed scheduling method. Instead of the greedy approach considered in their previous work, in this work Lei et al. proposed OL-PICEA-g, an offline multi-objective evolutionary algorithm based on an enhanced version of the PICEA-g algorithm (Wang et al., 2013). Experimental evaluation is performed over 10 different instances, each comprised by 1000 tasks, and the comparison is measured using the Two Set Coverage multi-objective metric (Coello Coello et al., 2007). Results show the proposed OL-PICEA-g is accurate and outperforms PICEA-g when addressing the proposed problem. In average, only 13% of the solutions computed by OL-PICEA-g are dominated by the solution computed by PICEA-g. However, 39% of the solutions computed by PICEA-g are dominated by the solutions computed by OL-PICEA-g.

2.5 Summary

The main challenges for optimizing energy efficiency in data centers consist in reducing energy consumption of computing elements and cooling systems, and incorporating renewable energy sources. All these challenges must be simultaneously addressed for maximizing energy efficiency.

Optimizing the energy consumption of computing elements is certainly the most studied of the presented challenges. Most approaches for optimizing it are based on dynamic power management features such as DVFS, low-power state modes, and per-core gating. However, few works consider characteristics such as networking, partial server load, or multi-core architectures. Table 2.1 presents a summary of the most recent research dealing with optimizing the energy consumption of computing elements.

Cooling systems have proved to be the second larger energy-consuming components in modern data centers. As such, these systems have become the target of many different energy optimization approaches. Because of the high complexity of accurately modeling heat dissipation and air circulation in data centers, thermal models simplify reality aiming at reducing the computing cost of the model. However, there is little consensus on the most appropriate approach for thermal modeling. Hence, several thermal models for data centers are available in the literature. Table 2.2 presents a summary of the most recent research dealing with optimizing cooling systems.

Finally, incorporating renewable energy sources has become a major trend

Table 2.1: Summary of works dealing with energy efficiency of computing elements.

Work	Energy efficiency metric	Energy model	Scheduling approach
Single server			
Wang et al. (2011)	Min. energy	DPM	Online
Group of servers			
Von Kistowski et al. (2015)	Min. energy	DPM	Online
Single data center			
Mei et al. (2013)	Min. energy	Low-power state	Offline
Shi et al. (2017)	Energy constrained	Server load	Online
Garg and Singh (2016)	Min. energy	DVFS	Offline
Chen et al. (2016a)	Min. energy	DVFS	Offline
Tang et al. (2016)	Min. energy	DVFS	Offline
Xie et al. (2016)	Energy constrained	DVFS	Offline
Sajid and Raza (2017)	Min. energy	DVFS	Online
Federation of data centers			
Guo and Fang (2013)	Min. budget	Server load	Online
Jena (2015)	Min. energy	Low-power state	Offline
Kaushik and Vidyarthi (2016)	Min. energy	Low-power state	Offline

Table 2.2: Summary of works dealing with energy efficiency of cooling systems.

Work	Energy efficiency metric	Thermal model	Scheduling approach
Group of servers			
Wang et al. (2010)	Min. energy	Wang et al. (2010)	Online
Single data center			
Al-Qawasmeh et al. (2015)	Min. energy & Energy constrained	Tang et al. (2006)	Online
Lee et al. (2017)	Min. energy	Chandra et al. (2002) & Moore et al. (2005)	Online
Meng et al. (2015)	Min. energy	Tang et al. (2006)	Online
Habibi Khalaj et al. (2015)	Min. hot spots	Habibi Khalaj et al. (2015)	Offline
Manousakis et al. (2015)	Min. cost	Regression-based	Online
Goudarzi and Pedram (2016)	Min. budget	Tang et al. (2006)	Online
Lin and Deng (2017)	Min. energy	Tang et al. (2006)	Online & Offline
Federation of data centers			
Polverini et al. (2014)	Min. budget	Mukherjee et al. (2009)	Online
Rajabi et al. (2014)	Max. profit	Linear model	Online
Xu et al. (2015)	Min. energy	Regression-based	Online

Table 2.3: Summary of works dealing renewable energy sources.

Work	Energy efficiency metric	Renewable energy prediction model	Scheduling approach
Single data center			
Ghamkhari and Mohsenian-Rad (2013)	Min. budget	Reactive	Online
Lei et al. (2015)	Min. budget	Synthetic noise	Online
Lei et al. (2016)	Max. renewable energy	Synthetic noise	Offline
Chen et al. (2016b)	Min. budget	Reactive	Online
Federation of data centers			
Abbasi et al. (2014)	Min. budget	Regression-based	Online
Yu et al. (2015)	Min. budget	Damousis et al. (2004)	Online
Paul et al. (2016)	Min. budget & Carbon emissions	Reactive	Online
Chen et al. (2016c)	Min. budget	Zhang et al. (2013)	Online
Kiani and Ansari (2016)	Max. profit	Reactive	Online
Anastasopoulos et al. (2016)	Max. renewable energy	Regression-based	Online
Peng et al. (2017)	Min. budget	Moving average	Online
Toosi et al. (2017)	Min. energy	Reactive	Online

in energy efficient data centers. This challenge gained popularity only recently but rapidly became a hot topic. The major challenge in incorporating renewable energy sources in general purpose data centers lie in accurately and robustly forecasting energy availability. Again, as with the thermal model, there is little consensus on the most appropriate forecasting model. Table 2.3 presents a summary of the most recent research dealing with optimizing the renewable energy usage.

Optimizing energy efficiency in data centers usually impacts the system performance, hence considering a QoS-related metric in the problem formulation is key for asserting the feasibility of the proposed method. Without a doubt, energy efficiency in data centers is a complex multi-objective problem. Many authors simplify the formulation of the energy efficiency problem by considering QoS as a problem restriction and defining a minimum QoS threshold. However, this approach fails to depict the true relation between objectives. A multi-objective optimization approach with a Pareto-based analysis is required for studying the relation between both objectives and harnessing the most adequate balance between energy efficiency and performance.

This thesis proposes a multi-objective approach for dealing with the problem of energy efficiency in data centers, considering the QoS of the system. Several problems are addressed, taking into consideration modern multi-core

architectures, an accurate thermal model, forecasting of renewable energy generation, and geographic location. Furthermore, results are studied using a Pareto-based multi-objective approach, analyzing the trade-off between the different optimization objectives.

Chapter 3

Scheduling Energy Efficient Data Centers Using Renewable Energy

Santiago Iturriaga and Sergio Nesmachnow

Abstract: This work presents a multi-objective approach for scheduling energy consumption in data centers considering traditional and green energy data sources. This problem is addressed as a whole by simultaneously scheduling the state of the servers and the cooling devices, and by scheduling the workload of the data center, which is comprised of a set of independent tasks with due dates. Its goal is to simultaneously minimize the energy consumption budget of the data center, the energy consumption deviation from a reference profile, and the amount of tasks whose due dates are violated. Two multi-objective evolutionary algorithms hybridized with a greedy heuristic are proposed and are enhanced by applying simulated annealing for post hoc optimization. Experimental results show that these methods are able to reduce energy consumption budget by about 60% while adequately following a power consumption profile and providing a high quality of service. These results confirm the effectiveness of the proposed algorithmic approach and the usefulness of green energy sources for data center infrastructures.

3.1 Introduction

Energy consumption in data centers has become a critical matter, especially to large providers like Google, Facebook, and Amazon among others. The Internet industry requires more and bigger data centers to sustain its growth, but the amount of energy required for their operation has become an issue for economic and environmental reasons. In 2005, data centers accounted for around 0.5% of the total world energy consumption, while in the year 2010, it reached an estimate of 1.3%. With an estimated annual growth rate of more than 16%, forecasts are grim (Koomey, 2008, 2011). By the year 2030, the worst-case scenario estimates the electricity consumption from data centers will reach up to 13% of the total world energy (Andrae and Edler, 2015).

Significant efforts have been made to address the aforesaid energy issues with different approaches and techniques (Dayarathna et al., 2016; Zhang et al., 2016). Many traditional energy saving methods focus on lowering energy consumption when computing resources are idle, or reducing the system performance when executing non-critical computing tasks, or scheduling the computational load when energy is cheaper (Nesmachnow et al., 2013; Dorronsoro et al., 2014a; Goiri et al., 2015a; Lei et al., 2016). These methods are helpful, but their effectiveness is limited because they usually lower the quality of service delivered by the data center. More recently, advances in renewable energy sources—known as green energy—have provided new ways to reduce energy consumption and the environmental impact of data centers (Wang et al., 2013). The use of green energy sources has empowered energy saving methods, enabling the scheduling of computational workload when green energy is available. Nevertheless, this remains a challenging approach since green energy sources are unreliable and depend on uncontrollable external conditions such as sun irradiation and wind velocity. Because of their unreliability, most data centers must consider a hybrid energy approach, with green energy combined with traditional energy—known as brown energy—to avoid energy outages.

When optimizing energy consumption in data centers, most approaches focus on optimizing *Central Processing Units* (CPUs) and cooling devices that are the most power hungry components in data centers. CPUs are, by far, the most power hungry components, using an estimated 46% of the total power consumption of a data center. The second most power hungry components are cooling devices, such as air conditioning systems, which consume an estimated

of 15% of the total power. Despite the fact that cooling devices consume less energy than CPUs, they are a key focus when addressing energy consumption since they are the most inefficient components and the best candidates for improvement (Barroso et al., 2013).

This work presents a multi-objective problem formulation for optimizing the energy consumption of data centers that are powered by hybrid energy. The problem formulation addresses the scheduling of computational load and cooling devices in a data center in order to minimize its energy consumption budget, minimize the deviation of its energy consumption from a reference consumption profile, and maximize the Quality of the Service (QoS) provided to its users. Two Multi-Objective Evolutionary Algorithms (MOEAs) are applied for effectively solving the scheduling problem: the Non-dominated Sorting Genetic Algorithm, version II (NSGA-II) (Deb et al., 2002) and the Epsilon-Variable Multi-Objective Genetic Algorithm (ev-MOGA) (Herrero et al., 2014). Experimental results show the proposed approach reduces energy consumption budget by about 60% while maintaining QoS over 95% and a deviation from a reference power profile of about 3%, all this compared to a traditional business as the usual data center scenario.

This work is an extension of our previous work (Iturriaga and Nesmachnow, 2015). This work further extends it by applying ev-MOGA and comparing its accuracy with NSGA-II when addressing the proposed problem. In addition to this, a larger and more diverse set of problem instances are evaluated and a more thorough multi-objective experimental analysis with statistical support is provided.

This article is organized as follows: Section 3.2 briefly reviews the most relevant and recently published work. Section 3.3 defines the scheme of the data center model. Section 3.4 presents the problem formulation. Section 3.5 describes the proposed MOEA designed for solving the problem. Section 3.6 reports and discusses the results of the experimental evaluation. Finally, Section 3.8 presents conclusions and future work.

3.2 Related Work

Many articles in the literature address the energy efficiency problem in data centers. Next, some of the latest and most relevant ones are presented.

Goiri et al. (2015a) propose GreenSlot, a single-objective greedy heuristic for scheduling computational load in data centers powered by green and brown energy. The aim of GreenSlot is to schedule a set of independent parallel computing tasks using the minimum amount of traditional energy while also meeting all task deadlines. Although GreenSlot takes into consideration a hybrid energy source, it uses brown energy just as a backup for when it cannot meet all deadlines using green energy alone. Furthermore, Goiri at al. use a realistic energy forecast model based on historical data to predict the availability of green energy. Goriri et al. show GreenSlot can reduce energy cost by up to 39% when compared to conventional schedulers. The work from Goriri et al. is different from ours mainly in that they address a single-objective problem considering QoS as a constraint, and because they do not consider cooling devices as energy consuming components nor temperature as an operative constraint.

Lei et al. (2016) address the energy efficiency problem in data centers by scheduling a set of independent tasks with deadlines to be executed in a system with Dynamic Voltage Frequency Scaling (DVFS) and powered by hybrid energy. The scheduler uses DVFS for reducing energy consumption when executing tasks with loose deadlines. In this formulation, a task can be rejected when its deadline cannot be met. Lei et al. formulate a multi-objective problem and propose two MOEA for solving it, optimizing four objectives: (i) maximizing green energy usage; (ii) minimizing the finishing time of the whole schedule; (iii) minimizing the total energy consumption; and (iv) minimizing the task rejection rate. Then, the proposed MOEAs are compared with each other using a simple multi-objective metric. Although Lei et al. argue that these results confirm the effectiveness of their approach, the amount of energy consumption saved when using it is not clear. This work is different from the work of Lei et al. in that, in our work, the temperature of the data center is a constraint and cooling devices are considered when computing the energy consumption. In addition, a thorough multi-objective analysis is performed in our work and statistical confidence results are presented.

Peng et al. (2017) deal with the scheduling of resources in multiple data centers powered by hybrid energy. The aim of this problem is to allocate virtual machines to physical machines in order to satisfy the computing demand while minimizing total energy cost and considering quality of service as a constraint. Peng et al. propose a single-objective formulation for addressing this problem and present an evolutionary algorithm for solving it. Peng et al. report that

this approach reduces energy cost between 8% and 10% when compared to traditional allocation schemes. Once more, the work from Peng et al. is different from ours in that, in our work, a multi-objective problem is addressed and the scheduling of cooling devices and the temperature of the data center as a constraint are both considered.

In our previous work (Iturriaga and Nesmachnow, 2015), a data center powered by hybrid energy was modeled, considering cooling devices and internal data center temperature. The multi-objective scheduling problem of minimizing the energy budget, maximizing the QoS, and meeting a brown energy consumption threshold is addressed, constrained to the operative temperature of the data center. The scheduling problem consisted of scheduling a set of independent computing tasks with due dates into a set of computing resources. An improved scheduling algorithm is proposed by hybridizing NSGA-II with a Local Search (LS) operator in order to exploit the most promising solutions. Experimental results show that this approach is able to reduce the energy budget by about 30% when compared to traditional schedulers. This work extends our previous work by comparing evMOGA—a novel state of the art MOEA—with NSGA-II, our previously proposed approach. The experimental analysis is improved by increasing the number and diversity of evaluated problem instances and by performing a more thorough experimental analysis presenting multi-objective metrics, providing statistical confidence.

3.3 The Data Center Energy- and QoS-Aware Model

The proposed data center model is based on the one proposed by Nesmachnow et al. (2015) and further extended by Iturriaga and Nesmachnow (2015). Figure 3.1 shows the schema of the data center model.

This model is comprised of two main components: the Heating, Ventilation and Air Conditioning (HVAC) component and the Computing Infrastructure (CI) component. The HVAC component represents the cooling devices, while the CI component represents the computing resources of the data center (i.e., the servers). A simple but realistic energy model is considered where each server may be either in the executing state, in the idle state, or in the sleep state. Every time a server executes a task, it is considered to be at peak

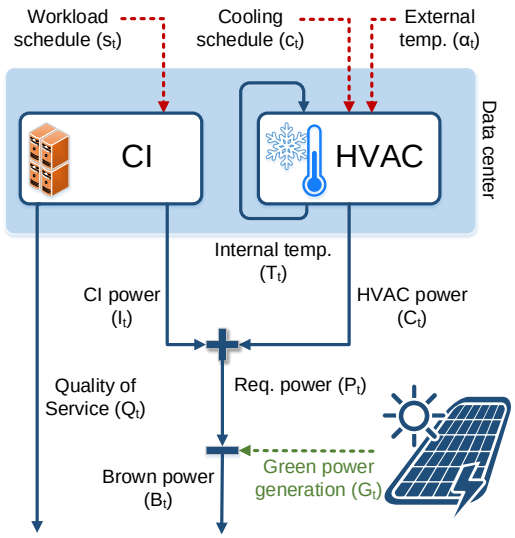


Figure 3.1: Data center scheme considering energy consumption and quality of service.

performance, thus consuming the maximum amount of energy for the server. When a server is on but not executing a task, it is considered to be idle and consuming a lower amount of energy. Finally, a server may be in sleep state and consuming a minimum amount of energy. Formally, the total energy consumption of the CI component (I_t) is calculated as $I_t = S_t^{max} + S_t^{idle} + S_t^{sleep}$, where S_t^{max} is the total energy consumption of servers that are executing a task, S_t^{idle} of which are in idle state, and S_t^{sleep} of which are in sleep state.

The data center is controlled by two input variables: task schedule (s_t) and cooling schedule (c_t). The task schedule s_t defines the allocation of tasks to computing resources in the CI component for each time unit. Likewise, the cooling schedule c_t defines the cooling strategy of the HVAC component for each time unit.

Furthermore, two non-controllable input variables are considered: external temperature (α_t) and green energy availability (G_t). These variables are external and are not controllable by the scheduling algorithm. The external temperature variable, α_t, represents the temperature outside the data center at each time unit. This temperature affects the effectiveness of the free cooling method. While the green energy availability variable, G_t, represents the amount of energy generated by the renewable energy source for each time unit.

Finally, three output variables are defined for the model: (i) the amount of overdue time required for task completion (Q_t); (ii) the internal temperature

variable (T_t) which is the temperature inside the data center; and (iii) C_t which is the energy required by the HVAC component, and I_t which is the energy required by the CI component. Hence, the total power required by the data center (P_t) is the sum of the power required by the HVAC and the CI components. Similarly, the total amount of traditional energy required by the data center (B_t) is the difference between the required power (P_t) and the amount of green power available (G_t).

The computing workload of the data center is comprised of a set of independent tasks with due dates. For a task, its due date is its expected or desired completion time. In this approach, the QoS of the system is measured by the total amount of additional time, over the due date, required by the system to complete the execution of all tasks. It is key to consider a QoS-related objective when reducing energy consumption since the latter, most likely, will affect the QoS provided to the final users.

The variable c_t schedules the HVAC while considering two cooling methods: air conditioning and free cooling. Both of these methods are widely used cooling methods (Barroso et al., 2013). The air conditioning mode uses the conventional Computer Room Air Conditioning (CRAC) unit which can take on and off values. The free cooling mode uses a fan system to inject air from the outside into the data center, and it can take fan speed values between 1 and 100. Hence, the energy consumption of the HVAC component at time t (C_t) is defined as shown in Equation (3.1):

$$C_t = \begin{cases} CompressorPWR, & \text{if air conditioning mode is on,} \\ FanPWR(s), & \text{if free cooling mode is on at speed } s \ (1 \leq s \leq 100), \\ 0, & \text{if both air conditioning and ventilation are off.} \end{cases} \tag{3.1}$$

The energy consumption of the air conditioning ($CompressorPWR$) is fixed, and $FanPWR(s)$ is between 0 and the maximum fan energy consumption. Because the cooling mode directly affects the temperature T_t in the data center, the Auto-Regressive eXogenous (ARX) model presented by Nesmachnow et al. (2014b) is used for modeling T_t. This model estimates the internal temperature of the data center taking into account the air conditioning, fan speed, outside temperature, and data center load.

3.4 The Problem Formulation

The data center must execute N tasks in a time horizon of K time units. The problem formulation consists of finding a schedule of computing tasks (s_t) and cooling resources (c_t) such that: it minimizes the difference between the energy consumption of the data center (P_t) with respect to a predefined reference profile (R_t), minimizes the energy budget, and minimizes the total overdue time of the tasks. Furthermore, the problem is subject to the internal temperature of the data center (T_t) which must be kept below its maximum operative value ($\hat{T}$).

Each task is an atomic unit which cannot be interrupted; once started, it must be executed to completion. Moreover, each task i should be completed before its due date $D(i)$ for the data center to deliver the optimum QoS level. With $FT(i)$ representing the finishing time of task i and M_t^b representing the brown energy cost at time t, optimization goals of this problem can be formally defined as presented in Equation (3.2):(3.3) and (3.4).

$$min\ z_p = \sum_{t=1}^{K} \begin{cases} (P_t - R_t)/\max(R_t), & \text{if } P_t > R_t, \\ 0, & \text{if } P_t \leq R_t, \end{cases} \tag{3.2}$$

$$min\ z_b = \sum_{t=1}^{K} B_t \times M_t^b, \tag{3.3}$$

$$min\ z_q = \sum_{i=1}^{N} \begin{cases} FT(i) - D(i), & \text{if } FT(i) > D(i), \\ 0, & \text{if } FT(i) \leq D(i). \end{cases} \tag{3.4}$$

Equation (3.2) defines the objective of minimizing the deviation of the power consumption of the data center with respect to the reference power profile R_t. Equation (3.3) defines the objective of minimizing the total monetary cost of the energy consumption of the system. Finally, Equation (3.4) defines the objective of minimizing the total overdue time required for completing the execution of all tasks.

3.5 Multi-Objective Evolutionary Scheduling for Energy-Aware Data Centers

Evolutionary algorithms are iterative stochastic optimization methods inspired in the evolution of the species and the natural selection process (Yu and Gen, 2012). Algorithm 1 presents the general schema of an evolutionary algorithm. At each iteration, the evolutionary algorithm stochastically applies a set of evolutionary operators to a population of candidate solutions to the optimization problem. These evolutionary operators combine and mutate solutions in the population, producing new solutions that compete with each other to survive the selection process and remain in the population. The recombination operator exploits known solutions by producing new solutions with the best characteristics of solutions already in the population. On the other hand, the mutation operator explores the solution landscape searching for solutions with new characteristics by applying small perturbations to known solutions. These methods have been successfully applied for solving optimization, search, and learning problems in many application domains.

Multi-Objective Evolutionary Algorithms (MOEAs) are evolutionary algorithms that deal with more than one simultaneous optimization objective. Unlike single objective optimization methods, MOEAs are able to compute a set of trade-off solutions in a single execution thanks to their population-based approach. However, an MOEA must maintain a adequate balance between optimizing each objective as much as possible, and sampling a diverse and homogeneously distributed Pareto front.

In this work, NSGA-II (Deb, 2001) and ev-MOGA (Martínez-Iranzo et al., 2009) are applied for addressing the proposed scheduling problem. These MOEA are entrusted with planning the state of servers and the cooling devices. On top of these MOEA, two hybridization mechanisms are further applied. The first mechanism consists of applying a greedy scheduling heuristic during the evolutionary process for reducing the dimension of the problem search space. This greedy heuristic receives a server and cooling schedule, and computes a scheduling for the task workload. The second mechanism consists of applying Simulated Annealing (SA) as a post hoc optimization procedure. That is, once the evolutionary process has finished, the SA is applied for further improving the accuracy of the solutions computed by the MOEAs. Next, the proposed algorithmic approach is described in detail.

Algorithm 1 General schema of an evolutionary algorithm

$g \leftarrow 0$

$P^0 \leftarrow initializePopulation()$ ▷ create the initial population of parents

$F_p^0 \leftarrow evaluate(P^0)$ ▷ evaluate population of parents

while stopping condition is not met **do**

$\lambda^g \leftarrow selectParents(P^g)$ ▷ select parents to evolve

$\mu^g \leftarrow recombine(\lambda^g)$ ▷ recombine parents and create offspring

$\mu^g \leftarrow mutate(\mu^g)$ ▷ mutate offspring

$F_\mu^g \leftarrow evaluate(\mu^g)$ ▷ evaluate population of offspring

$P^{g+1} \leftarrow createNewPopulation(P^g, F_p^g, \mu^g, F_\mu^g)$ ▷ new population with the best individuals

$g \leftarrow g + 1$

end while

3.5.1 Solution Representation

For representing the problem solution, a time discretization approach is applied using a time horizon of $K = 150$ time steps. For each time step, the solution encodes the number of active servers and the state of the cooling devices.

The active servers are encoded directly with their total power consumption in Watts. Hence, an integer vector of size K is used, with values ranging from 0 up to S^{max}, where S^{max} is the power consumed when all servers are active. Similarly, the state of the cooling devices is encoded as an integer vector of size K with each value representing three states in the interval $[1, 300]$. A value v in the interval $[1, 100]$ represents the free cooling being used with a fan speed of v. On the other hand, a value v in the interval $[101, 200]$ represents the CRAC being used. Finally, a value v in the interval $[201, 300]$ means that all cooling devices are off.

Considering both encodings, the complete solution representation is an integer vector of size $2K$ where the cooling devices are encoded in the first K integers of the representation, and the server state is encoded in the last K elements of the representation. This representation was previously introduced by Nesmachnow et al. (2014b). Figure 3.2 shows the representation of a sample solution.

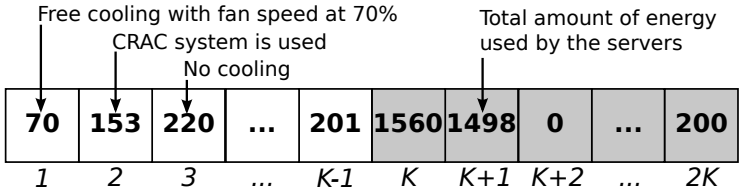


Figure 3.2: Representation of a sample solution.

3.5.2 Initial Population

Each solution of the initial population is created randomly using the following criteria. With probability $p = 0.5$, the new solution is created totally at random. That is, each value of the solution encoding is chosen randomly following a uniform distribution in the whole range of valid values. Otherwise, the new solution is created randomly in a high energy subspace of the solution space. For this, each value of the encoding is chosen in the upper 20% of the range of valid values. This technique allows us to provide the MOEA with a greater number of feasible solutions in the initial population.

3.5.3 Evolutionary Operators for NSGA-II

NSGA-II is a well-known MOEA proposed by Deb (2001). In this work, the NSGA-II implementation previously proposed by Iturriaga and Nesmachnow (2015) is applied for addressing the scheduling problem since it proved to be accurate and efficient. Next, the recombination and mutation operators for the NSGA-II are presented.

Recombination Operator

A three-point operator is defined for the recombination operator. This operator starts by selecting three points: p_1, p_2, p_3. Point p_1 is selected randomly following a uniform distribution in the interval $[1, K-1]$. Point p_2 is always K, and point p_3 is $K + p_1$. By defining p_2 to always be K, it is asserted that cooling and server power values at each time step from each parent are inherited together by the offspring. For example, the value at position $1'$ is always inherited with position $K+1'$, each corresponding to cooling and server power values at time step 1. Figure 3.3 shows how this operator works.

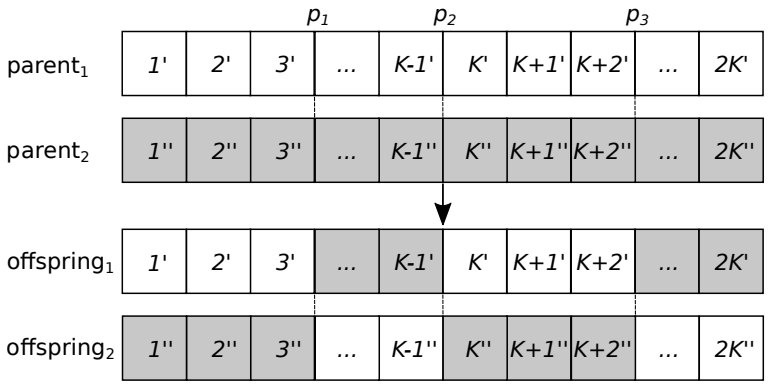


Figure 3.3: Example of the three point recombination operator for the Non-dominated Sorting Genetic Algorithm, version II (NSGA-II).

Mutation Operator

The mutation operator is applied individually to each integer in the solution encoding but uses two different approaches, one for the cooling state part of the solution and another for the server power part. For the cooling state part (i.e., the first K elements in the encoding), a differential approach is applied. That is, for mutating an integer value v, a random value $r \in [0, 1)$ is selected, and the mutated value of v is computed as $v' = (v + r \times 300) \bmod 300$. For the server power part of the encoding (i.e., elements $K+1$ to $2K$), a simple uniform mutation is applied, such that the value of v' is a random value between 0 (i.e., when all servers are off) and S^{max} (i.e., when all servers are on).

3.5.4 Evolutionary Operators for ev-MOGA

The ev-MOGA was proposed by Martínez-Iranzo et al. (2009) to address the shortcomings of many MOEAs. In this work, ev-MOGA uses the exact same mutation operator as NSGA-II. This mutation operation proved to be useful and adapted adequately for ev-MOGA. On the other hand, a linear recombination is used for the recombination of an operator, as recommended by the original ev-MOGA implementation. This latter operator is detailed next.

Recombination Operator

For the recombination operation, a linear recombination method is applied such that parents P_1 and P_2 are recombined to produce offspring O_1 and O_2 as shown in Equation (3.5) and (3.6):

$$O_1 = \alpha \otimes P_1 + (1 - \alpha) \otimes P_2, \tag{3.5}$$

$$O_2 = (1 - \alpha) \otimes P_1 + \alpha \otimes P_2, \tag{3.6}$$

where α is a random vector in the space $[0, 1)^{2K}$ and $\otimes$ is the element-wise multiplication of two vectors. Finally, all values of O_1 and O_2 are rounded to their nearest integer value to satisfy the encoding.

Fitness Functions

The fitness functions that guide the evolutionary process are identical to the ones defined in Equation (3.2), (3.3) and (3.4). Because the MOEA solution encodes the cooling and server power, it alone is sufficient for computing Equation (3.2) and (3.3). However, because it does not include task scheduling information, it is not possible to compute Equation (3.4).

For computing Equation (3.4), the Best Fit Hole (BFH) algorithm is used as a subordinate greedy scheduling heuristic for computing a task-to-server scheduling. Next, the BFH task-scheduling heuristic is presented in detail.

Greedy Task Scheduling Heuristic

The Best Fit Hole (BFH) heuristic is applied for task scheduling (Dorronsoro et al., 2014a). This heuristic works by applying a backfilling approach and keeping track of holes in the schedule when servers are idle. Tasks are assigned, one by one, to the hole that satisfies their due date and minimizes the difference between its execution time and the hole duration (i.e., it best fits the task). If no hole satisfies the deadline constraint of a task, then that task is scheduled at the end of the server that can execute it faster. Algorithm 2 shows the schema of the BFH scheduler.

3.5.5 Simulated Annealing for Post Hoc Optimization

Simulated Annealing (SA) is a metaheuristic introduced by Kirkpatrick et al. (1983). In this work, it is applied as a post hoc optimization for further improving the solutions computed by the MOEA. A similar post hoc approach proved to be very effective in significantly improving QoS by Iturriaga and Nesmachnow (2015).

Algorithm 2 Schema of the Best Fit Hole (BFH) scheduling heuristic

$U \leftarrow tasks$ ▷ queue of tasks

$H \leftarrow \emptyset$ ▷ set of holes

while $U \neq \emptyset$ **do**

$t \leftarrow first(U)$ ▷ get the first task and remove it from the set

$H^t \leftarrow canAcomodateTask(H, t)$ ▷ get holes which may accommodate the task

$H^t \leftarrow satisfiesDueDate(H^t, t)$ ▷ get holes which satisfy the due date of the task

if $H^t \neq \emptyset$ **then**

$h \leftarrow getSmallestHole(H^t)$ ▷ get hole with smallest length

$assignTaskToHole(t, h)$

$H = (H \setminus h) \cup (h \setminus t)$ ▷ remove hole from set and update remaining hole time

else

$assignToFastestServer(t)$

end if

end while

Since SA works with a single solution, it is executed separately from each non-dominated solution computed by the MOEA. On top of that, SA is a single objective optimization method and cannot deal with multiple objectives. For this reason, it is applied multiple times (N) for each solution. Each time, the SA randomly chooses an optimization objective to be improved. Algorithm 3 shows the schema for applying the post hoc optimization. Dominance is used as an acceptance criterion, that is, a new solution is accepted only when it dominates the current solution. The neighborhood for the SA is constructed using a simple task moving operation which moves one task from its current machine to a new machine in some arbitrary position.

Algorithm 3 Schema for post hoc optimization using Simulated Annealing (SA)

```
S ← solutions                          ▷ set of solutions computed by the MOEA
S* ← ∅                                 ▷ set of improved solutions
while S ≠ ∅ do
    s ← first(S)
    for i = 1 : N do
        s' ← applySA(s)                ▷ applies simulated annealing to s
        if s' dominates s then         ▷ evaluates Pareto dominance
            s ← s'
        end if
    end for
    S* = S* ∪ s
end while
```

3.6 Experimental Evaluation

In this section, the experimental evaluation of the proposed algorithmic solutions is presented. It introduces the problem instances created for assessing the efficacy of the proposed algorithms and the parameter setting of each algorithm.

3.6.1 Problem Instances

A set of realistic problem instances is created to evaluate the proposed algorithmic approach. For all these problem instances, a scheduling horizon of $K = 150$ time steps is defined and a rescheduling policy is considered. Every K time steps or when a new task arrives, the scheduling algorithm is executed and all queued tasks are rescheduled. This is a realistic time horizon with an adequate compromise of time ahead scheduling and rescheduling frequency. An external temperature of 25 ℃is considered, an initial internal temperature of 26.5 ℃, and a maximum operative temperature of 27 ℃. The data center simulates the Parasol data center presented by Goiri et al. (2015a). This data center consists of 64 low-power servers with Intel Atom processors. Each server consumes 30 W when executing at peak performance, 22 W in idle state, and only 3 W in sleep state. The CRAC system consumes 2.3 kW and the free cooling system up to 410 W.

Considering all the above, small, medium, and large workloads are defined comprised of 200, 300 and 400 tasks, respectively. Three different workload instances are created for each size totaling nine different workload instances. For the reference power profile, R_t, three different configurations are considered. These configurations represent a fairly heterogeneous spectrum of scenarios with varying reference profiles. Figure 3.4 shows all three reference power profiles. Regarding the green power generation, an array of three solar panels are simulated, each one capable of producing up to 0.5 kW. Three different green power profiles are considered for these solar panels. The first one, g_1, is shown in Figure 3.5a and represents a morning power generation profile. This profile represents a scenario that starts with low energy generation and gradually increases energy generation as midday approaches. Profile g_2, which corresponds to a midday power generation profile, is shown in Figure 3.5b. This profile represents the maximum energy generation profile with peak solar irradiation. Finally, profile g_3 equals a null green power generation representing a nighttime environment. This profile is not even shown, as no energy is generated during the night. All three of these power profiles were generated using historical solar irradiation information by Goiri et al. (2015a).

In total, 81 different instances are created considering a wide range of different scenarios.

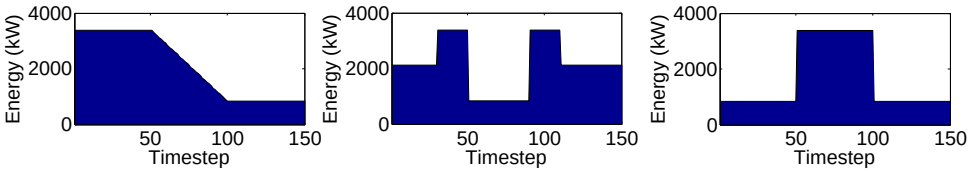


Figure 3.4: Power reference profiles. (**a**) profile A; (**b**) profile B; and (**c**) profile C.

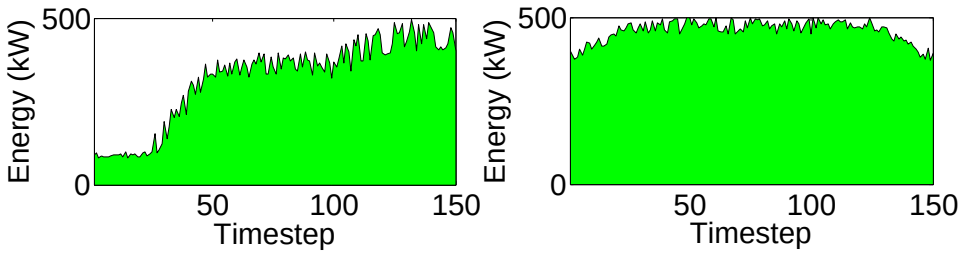


Figure 3.5: Green power generation profiles. (**a**) morning profile (g_1); and (**b**) midday profile (g_2).

3.6.2 Parameter Settings

In this work, NSGA-II is configured with a population size of $\#pop = 20$, a recombination probability of $p_r = 0.9$, and a mutation probability of $p_m = 0.01$. On the other hand, ev-MOGA is configured with a P population size of $\#pop_P = 100$ and a G population size of $\#pop_G = 8$. A mutation probability of $p_m = 0.1$ and a recombination probability of $p_r = 0.9$. These settings are heavily based on the values proposed originally by Deb (2001) and Martínez-Iranzo et al. (2009). Both MOEAs were configured with a stopping criterion (sc) of 1000 generations for keeping the algorithms within reasonable execution times.

As for the post hoc optimization, SA is configured to be executed $N = 12$ times for each non-dominated MOEA solution. The initial temperature of SA was configured as $T_0 = 1$, the stopping temperature (st) as $T_i \leq 1 \times 10^{-8}$, and the cooling schedule (cs) as $T_{i+1} = T_i \times 0.8$. These settings are heavily based on the values originally proposed by Kirkpatrick et al. (1983). Table 3.1 summarizes NSGA-II, ev-MOGA and SA settings.

Table 3.1: Parameter settings for Non-dominated Sorting Genetic Algorithm version II (NSGA-II), Epsilon-Variable Multi-Objective Genetic Algorithm (ev-MOGA) and simulated annealing (SA).

Algorithm	Parameter	Value
NSGA-II	p_r	0.9
	p_m	0.01
	$\#pop$	20
	sc	1000 generations
ev-MOGA	p_r	0.9
	p_m	0.1
	$\#pop_P$	100
	$\#pop_G$	8
	sc	1000 generations
SA	N	12
	T_0	1
	st	$T_i \leq 1 \times 10^{-8}$
	cs	$T_{i+1} = T_i \times 0.8$

3.7 Experimental Results and Discussion

This section presents and discusses the experimental results computed by both algorithmic approaches, the one based on NSGA-II and the one based on ev-MOGA. First, the efficacy of these approaches is studied by comparing their accuracy at optimizing the objectives and at sampling the Pareto front. After that, the best approach is compared with a traditional business-as-usual scenario where no power optimization is performed and no green power source is available. This business-as-usual scenario is a typical scenario in many small- and medium-sized data centers.

3.7.1 NSGA-II and ev-MOGA Comparison

The hypervolume metric is applied for comparing the NSGA-II and ev-MOGA approaches. The hypervolume metric was introduced by Zitzler and Thiele (Zitzler and Thiele, 1998) and is the most used metric for comparing multiobjective optimization results (Riquelme et al., 2015). This metric works by constructing a hypercube v_i for each solution i in the Pareto front Q. Each hypercube is defined with a reference point W and solution i as corners of the diagonal of the hypercube. The reference point W is arbitrary but may be easily constructed using the worst computed values for each objective function.

Table 3.2: Average and standard deviation of the relative hypervolume (RHV) computed by each algorithm for each workload size and green power generation profile.

Workload Size	Green Power Profile	Average RHV	
		ev-MOGA	**NSGA-II**
small	g_1	**0.89 ±0.19**	0.72 ± 0.12
	g_2	**0.93 ± 0.12**	0.73 ± 0.13
	g_3	**0.84 ± 0.23**	0.61 ± 0.16
medium	g_1	**0.91 ± 0.14**	0.69 ± 0.16
	g_2	**0.89 ± 0.16**	0.76 ± 0.15
	g_3	**0.83 ± 0.18**	0.63 ± 0.13
large	g_1	**0.79 ± 0.13**	0.62 ± 0.17
	g_2	**0.86 ± 0.10**	0.67 ± 0.15
	g_3	**0.75 ± 0.14**	0.57 ± 0.14

Note: statistical differences are shown in **bold** font ($p \leq 0.001$).

Finally, the hypervolume (HV) is computed as the union of all hypercubes such that a larger hypervolume is always preferable. Equation (3.7) shows how hypervolume is calculated:

$$HV = volume\left(\cup_{i=1}^{|Q|} v_i\right). \tag{3.7}$$

The hypervolume metric considers both the convergence and the diversity of a Pareto front providing an all-around quantitative value. In this work, the relative hypervolume metric is reported for readability. The relative hypervolume is computed as $RHV = \frac{HV}{HV^*}$, where HV^* is the hypervolume of the best known approximation to the true Pareto front.

For the comparison study, a total of 30 independent executions were performed for each algorithm and for each problem instance. Table 3.2 shows the average and standard deviation of the relative hypervolume computed by each algorithm grouped by workload size and by green power profile. The Kruskal–Wallis test (Kruskal and Wallis, 1952) is used to provide statistical confidence in the comparison. Statistically significant differences are presented in **bold** font.

Overall results show ev-MOGA computes consistently better solutions than NSGA-II for all scenarios. Figures 3.6 and 3.7 present box plots for RHV results aggregated by dimension and by green power profile. For all scenarios, ev-MOGA presents a higher median RHV and lower dispersion values.

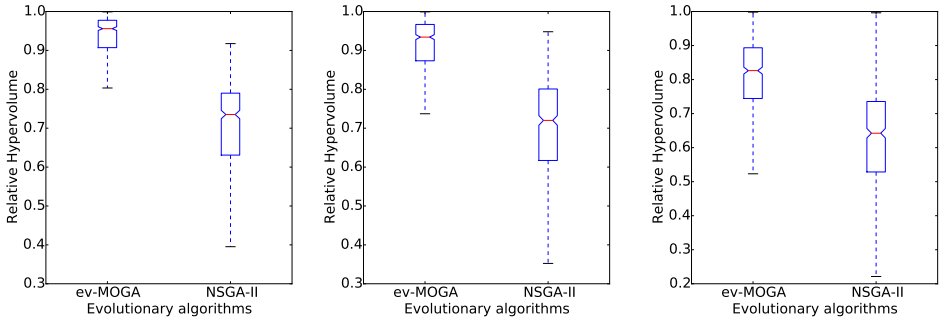


Figure 3.6: Relative hypervolume computed by Epsilon-Variable Multi-Objective Genetic Algorithm (ev-MOGA) and Non-dominated Sorting Genetic Algorithm version II (NSGA-II) for each workload size. (**a**) small workload size; (**b**) medium workload size; and (**c**) large workload size.

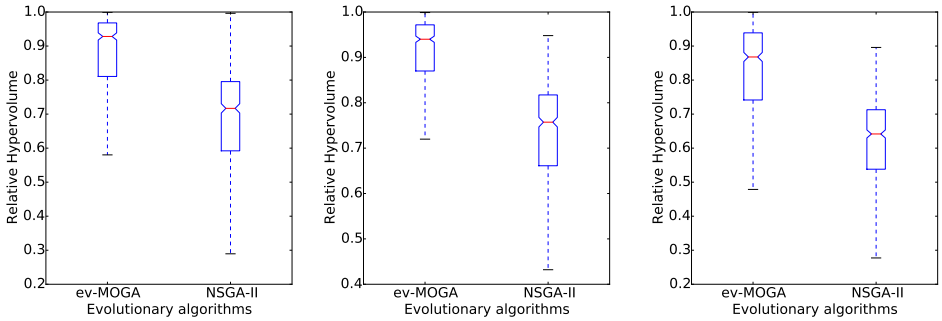


Figure 3.7: Relative hypervolume computed by Epsilon-Variable Multi-Objective Genetic Algorithm (ev-MOGA) and Non-dominated Sorting Genetic Algorithm version II (NSGA-II) for each green energy profile. (**a**) morning profile (g_1); (**b**) midday profile (g_2); and (**c**) night profile (g_3).

Next, the best aggregated results computed by each algorithm are analyzed, that is, the best approximation to the true Pareto front computed by each algorithm. Table 3.3 presents the best aggregated RHV values for each algorithm, for each workload size, and for each green power profile. Results show ev-MOGA is significantly better than NSGA-II for all problem instances with small- and medium-sized workloads. For small-sized workloads, ev-MOGA is always able to compute the best approximation to the true Pareto front without fail. Results are more contested for the large-sized workloads where ev-MOGA is significantly better than NSGA-II in just one out of three scenarios.

Figures 3.8a–c show a sample of the best aggregated Pareto front for each algorithm and for each green power profile for a medium-sized workload. In summary, ev-MOGA proved to be more accurate than NSGA-II in nearly all of the considered problem instances.

Table 3.3: Average and standard deviation of the relative hypervolume (RHV) for the best aggregated Pareto front computed by each algorithm for each workload size and green power generation profile.

Workload Size	Green Power Profile	Average RHV	
		ev-MOGA	**NSGA-II**
small	g_1	**1.00 ± 0.00**	0.87 ± 0.03
	g_2	**1.00 ± 0.00**	0.87 ± 0.03
	g_3	**1.00 ± 0.00**	0.82 ± 0.04
medium	g_1	**1.00 ± 0.00**	0.89 ± 0.04
	g_2	**0.99 ± 0.01**	0.91 ± 0.04
	g_3	**0.98 ± 0.02**	0.85 ± 0.04
large	g_1	0.97 ± 0.02	0.92 ± 0.06
	g_2	**0.99 ± 0.01**	0.91 ± 0.03
	g_3	0.93 ± 0.04	0.86 ± 0.04

Note: statistical differences are shown in **bold** font ($p \leq 0.001$)

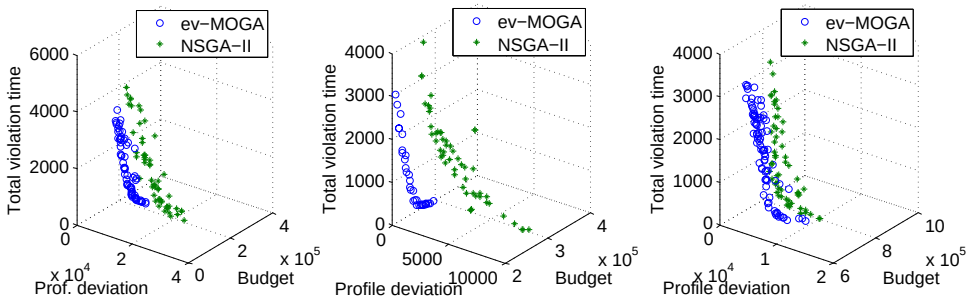


Figure 3.8: Best aggregated Pareto front computed by each algorithm for medium-sized workloads and for each green energy profile. (**a**) morning profile (g_1); (**b**) midday profile (g_2); and (**c**) night profile (g_3).

Table 3.4: Average and standard deviation for the relative budget reduction over the business-as-usual scenario (BAU), and relative deviation from the reference power computed by ev-MOGA for quality of service over 95%.

Workload Size	Green Power Profile	Budget Reduction over BAU	Relative Deviation from Ref. Power
small	g_1	76% ± 5%	1% ± 2%
	g_2	83% ± 3%	0% ± 1%
	g_3	58% ± 5%	2% ± 3%
medium	g_1	66% ± 4%	1% ± 2%
	g_2	70% ± 6%	1% ± 2%
	g_3	46% ± 5%	5% ± 4%
large	g_1	53% ± 2%	4% ± 3%
	g_2	59% ± 2%	1% ± 1%
	g_3	33% ± 3%	11% ± 5%

3.7.2 Comparison of ev-MOGA with the Business-as-Usual Approach

The business-as-usual scenario represents a real-world scenario where no green power source is available and where servers are never in sleep state. For this comparison, just solutions with a relative QoS of 95% or greater are considered. This is a realistic scenario since providing a low QoS is not desirable for most data centers. The relative QoS value is computed as the total amount of overdue time relative to the sum of execution time required by all tasks. Hence, a schedule with a relative QoS of 95% means it requires just 5% of additional overdue time for task execution.

Table 3.4 shows the average and standard deviation for the relative budget reduction computed by ev-MOGA over the business-as-usual scenario for each workload size and each green power profile with QoS of over 95%. On top of that, it also shows the average and standard deviation for the relative deviation from the reference power profile, with the deviation from the reference power profile being calculated as the total amount of energy required that surpasses the reference profile, relative to the total amount of energy specified in the reference profile.

Results show that ev-MOGA is able to compute a large budget reduction, up to 83%, with a small relative deviation from the power reference profile, lower than 5% in most scenarios. As expected, the larger reductions are com-

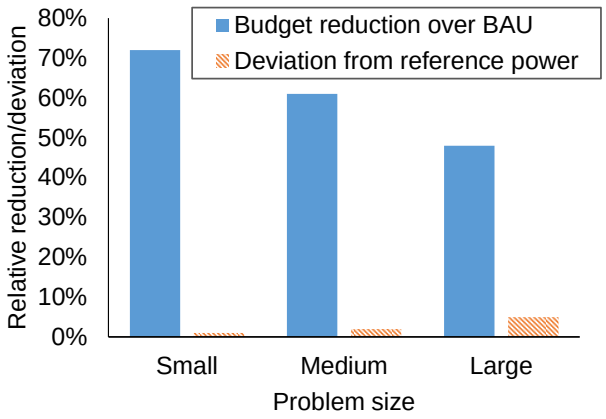


Figure 3.9: Average relative budget improvement over business-as-usual scenario and relative deviation from power reference computed by ev-MOGA for a relative quality of service over 95%

puted for the morning and midday green power profile (g_1 and g_2), with an average reduction of 68%. Budget reduction is much lower for the nighttime green power profile (g_3), with an average reduction of 46%, which accounts for just the server state and cooling device planning. These results show the effectiveness of considering green energy sources for data center infrastructures, reducing the energy budget by an average of 22%. Figure 3.9 shows the average budget reduction and relative power reference deviation for the solutions computed by ev-MOGA when compared to the business-as-usual scenario.

3.8 Conclusions

This work addresses the problem of controlling power consumption in data centers considering both traditional and renewable energy sources. This scheduling problem consists of simultaneously scheduling the states of the servers and the cooling devices of the data center, and the scheduling of tasks to be executed in the data center. A mathematical formulation is proposed for this problem, taking into account a desired reference power consumption profile, the overall electricity budget, and the QoS provided to its users.

A fully Pareto-oriented methodology is applied for solving this problem, and two multi-objective evolutionary algorithms—NSGA-II and ev-MOEA—are presented. Both are hybridized with a greedy scheduling algorithm and a simulated annealing algorithm. In the proposed schema, the multi-objective evolutionary algorithms schedule the server and cooling devices, while the

greedy heuristic schedules the execution of tasks. On top of that, simulated annealing is used as a post hoc optimization mechanism and it is executed at the very last moment.

A small low-power data center is modeled for the experimental analysis, taking into account the heat dissipation of its servers and the effectiveness of its cooling devices. A wide set of problem instances are constructed considering different power profiles, green power generation profiles, and workloads of tasks. First, NSGA-II and ev-MOEA approaches are compared using a Pareto-aware methodology to determine the best evolutionary approach. Next, the best approach is compared with a business-as-usual scenario with no scheduling of server or cooling devices, and with just a traditional energy source.

Results show the ev-MOGA approach is significantly more accurate than the NSGA-II approach for all of the problem instances, improving accuracy of the schedules by about 18% in average. When comparing ev-MOGA with the business-as-usual scenario, only schedules with high QoS values are considered, since low QoS values are certainly not desirable for most real-world data centers. In these high-QoS scenarios, ev-MOGA proved to be very effective, computing average budget reductions ranging from 33% up to 83%, depending on the workload size and green power generation profile. Furthermore, these reductions are obtained with an average deviation from the reference power profile of just about 3%.

These results confirm the effectiveness of the proposed approach and the usefulness of considering renewable energy sources for data center infrastructures. The main lines of future work include extending the experimental analysis and improving the efficiency of the proposed methods. The experimental analysis is to be improved by considering larger data center infrastructures and a wider range of green power generation profiles. At the same time, improving the efficiency and efficacy of the proposed methods is equally important and is certainly necessary in order to address larger problem instances.

Chapter 4

Multiobjective evolutionary algorithms for energy and service level scheduling in a federation of distributed datacenters

Santiago Iturriaga, Bernabé Dorronsoro, and Sergio Nesmachnow

Abstract: This article studies the application of multiobjective evolutionary algorithms for solving the energy-aware scheduling problem of workflows in a distributed system that is composed by a federation of datacenters. Nowadays, energy efficiency is of major concern when using large distributed computing systems, including the novel grid and cloud computing facilities. Researchers and system planners are looking for accurate methods to be used for planning the execution of large workloads that consume large amounts of resources, having a direct implicance in the energy consumption of the system and its operational costs. In the approach proposed in this article, we study the application of multiobjective evolutionary algorithms combined with low-level backfilling heuristics for finding efficient mappings of workflows into resources in order to maximize several metrics related to the quality of service, while reducing the energy required for computation. The experimental evaluation is performed considering both medium and large workloads that model realistic high performance computing applications and nowadays distributed comput-

ing infrastructures. The experimental results demonstrate that the proposed multiobjective evolutionary approaches compute accurate schedules, significantly outperforming both traditional round-robin/load-balancing schedulers and a set of combined list scheduling heuristics (accounting for both problem objectives) previously applied to the problem.

4.1 Introduction

Datacenters are large supercomputing facilities hosting many computing resources that provide multiple services, including computing power, networking, storage, etc. Nowadays, datacenters are being used in different application domains, most notably including science, industry and commerce (Khan and Zomaya, 2015).

The new paradigms for computation that propose using geographically distributed infrastructures to deal with large-scale complex problems (i.e. *grid* and *cloud* computing) have gained notorious interest due to the emergence of modern datacenter facilities and parallel computing methodologies and libraries. A common infrastructure for large-scale computing is a *federation* of distributed datacenters, i.e. a set of datacenters organized to work cooperatively as a single-image computing resource. Indeed, such a federation of datacenters provides a significantly large amount of computing power to be used in modern supercomputing applications. Each datacenter in a federation is typically composed by a large number of heterogeneous computational resources, including high performance clusters, large storage systems, and/or components of large grids or cloud systems (Zomaya and Lee, 2012).

Energy efficiency is a very relevant issue for datacenter operation and has become a major concern when using complex computing infrastructures (Ahmad and Ranka, 2012). Recent surveys reveal that datacenters and high performance computing facilities account for about 1.5% of the total world energy usage (Koomey, 2011), and for each datacenter, the electricity bills represent about 5% of the total cost of ownership (Barroso and Hölzle, 2009). As a consequence, datacenters' owners and operators are interested in maintaining the energy consumption as low as possible, for both economic and environmental reasons. However, the energy efficiency management is in conflict with the performance of the system, since increasing the performance requires using more energy, and reducing the energy consumption will negatively affect the

Quality of Service (QoS) that the datacenter provides to the users. This is a complex problem that requieres a multiobjective analysis for finding accurate solutions of the datacenter planning problem, providing different trade-offs between energy consumption and performance.

Different techniques for reducing the energy consumption in datacenters have been proposed, ranging from ad-hoc hardware solutions to more general software methods for specific infrastructures (Ahmad and Ranka, 2012; Dorronsoro et al., 2014a; Iturriaga and Nesmachnow, 2015; Nesmachnow et al., 2015; Tchernykh et al., 2015; Valentini et al., 2013; Khan and Zomaya, 2015). This article presents the application of MultiObjective Evolutionary Algorithms (MOEAs) for energy-aware scheduling of workloads with deadlines into a federation of distributed datacenters. Similar problems have been tackled by Abrishami et al. (2013) and Schwiegelshohn and Tchernykh (2012). The computing infrastructure considered in this work is composed by a number of clusters that might be geographically distributed. This is indeed the architecture of modern high performance computing systems, including supercomputers, high performance computing centers, and cloud infrastructures, among others. We specifically tackle scenarios with high performance computing applications which require high computing usage and low networking and storage usage. Thus, we approximate the energy consumed by each machine with the energy consumed by its central processing units.

A hierarchical two-level approach is applied (Dorronsoro et al., 2014a; Quezada-Pina et al., 2012), which divides the scheduling problem into a number of simpler and smaller sub-problems to be solved in each component of the datacenter federation. Scheduling heuristics and MOEAs are used to assign groups of worflows submitted by the users to each datacenter. Then, specific ad-hoc backfilling heuristics considering makespan, energy consumption, and QoS of solutions, are applied for scheduling within each datacenter.

This article is an extension of our previous conference paper, presented in the VIII ALIO/EURO Workshop on Applied Combinatorial Optimization (Montevideo, Uruguay, 2014), in which a hierarchical method was presented for energy-aware scheduling of large workloads into a distributed computing system (Nesmachnow et al., 2014a). That work contributed to the literature with a novel hierarchical model that uses two levels for assigning workflows to a number of geographically distributed resources, that allows scheduling a large number of workflows of tasks with dependencies.

The contributions of this article are: (*i*) a reformulation of the problem, including a user-oriented perspective by considering Service Level Agreement (SLA) contracts with customers over a heterogeneous federated datacenter; (*ii*) the resolution of the problem with versions of well-known scheduling heuristics adapted to the problem, and two new MOEAs specifically designed and implemented for the problem formulation proposed; (*iii*) the computation of new lower bounds for the problem, using a Constraint Programming (CP) methodology for estimation; and (*iv*) the use of large datacenters in the experimental analysis, the biggest one composed of over $1,500$ processing elements, a more realistic computational infrastructure than those in the instances used in our previous work.

We evaluated the studied schedulers with a set of 175 instances, including 50 medium size workloads—used to compute lower bounds for the problem objectives—and 125 workloads with large workflows that model typical high performance computing applications over realistic distributed infrastructures. As in our previous work (Dorronsoro et al., 2014a), five different classes of workloads are considered, namely Series-Parallel, Heterogeneous-Parallel, Homogeneous-Parallel, Single-Task, and Mix. The workloads of all problem instances contain 1000 workflows, with up to 132 tasks each, to be scheduled in a federation of datacenters with around $3,000$ computational units. Experimental results demonstrate that the MOEA schedulers compute accurate solutions with more than 37 solutions that outperform the best studied heuristic on all three objectives.

The article is organized as follows. The problem formulation and a review of the related work are presented in Section 4.2. The scheduling approach and the proposed multiobjective evolutionary methods are described in Section 4.3. The specific MOEAs studied to solve the problem are introduced in Section 4.4. The experimental evaluation is reported in Section 4.5, including a comparative analysis of the proposed schedulers and a comparison against both lower bounds computed for the problem and a traditional optimistic load balancing round robin scheduler. Finally, Section 4.6 formulates the conclusions of the research and the main lines for future work.

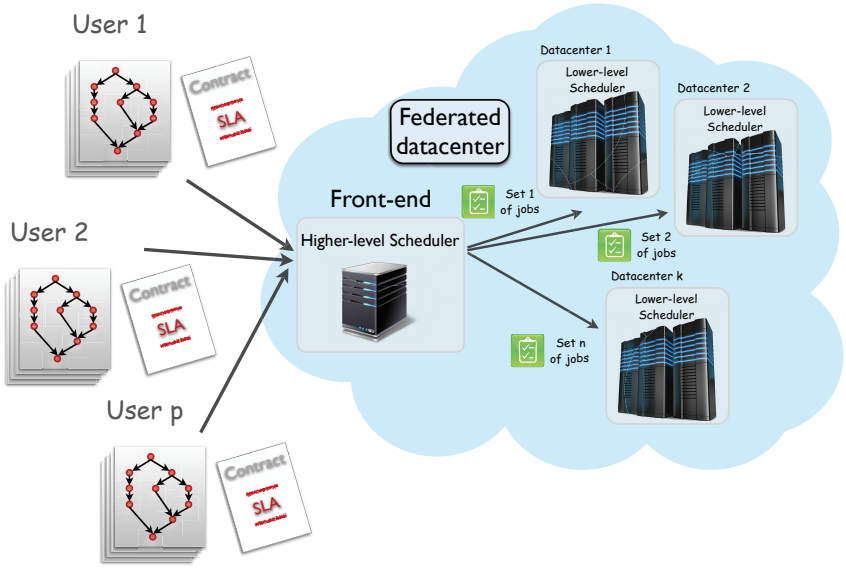


Figure 4.1: Overview of the scheduling problem in a federation of datacenters

4.2 The problem: energy-aware scheduling in a federation of datacenters

This section introduces the problem model and formulation, and discusses the related work about energy-aware scheduling in datacenters.

4.2.1 Problem model

The energy-aware scheduling problem addressed in this article is to find the best possible allocation of resources to execute a large number of parallel applications into a geographically distributed federated datacenter. We define the accuracy of an allocation in terms of the time required to execute all its applications (i.e., the makespan), its energy consumption, and its service level agreement violations. Figure 4.1 shows an overview of the problem. Our problem model is composed of the following elements:

- A distributed computing infrastructure (the federation of datacenters) formed by k heterogeneous datacenters $DC = \{dc_1, \ldots, dc_k\}$. Each datacenter DC_r is comprised of a set of multi-core machines (*servers*) $S_r = \{s_1, \ldots, s_s\}$, each machine s_j characterized by its number of cores c_j, performance in FLoating-point Operations Per Second (FLOPS) ops_j, and energy consumption of its processors at idle e_j^{idle} and peak e_j^{max} usage.

- A set of n independent heterogeneous workflows $Q = \{q_1, \ldots, q_n\}$. Each workflow q has an associated soft deadline d_q before it should be accomplished. Each workflow q is a parallel application that is decomposed into a (large) set of tasks $WT_q = \{wt_1, \ldots wt_m\}$ with dependencies among them. Typically, each task has different computing requirements.
- A number p of users (*owners*) $O = \{o_1, \ldots, o_p\}$, each having a number of workflows to be executed (defined by $wu(u_i)$) and an SLA agreement, determining the percentage of applications that should be finished before their deadlines. We consider three realistic levels of SLA agreements for large workload instances, i.e., $SLA = \{90\%, 94\%, 98\%\}$.
- Each task α is characterized by two values (o_α, nc_α) defining its length (number of operations), and the number of resources (cores) required for the parallel execution, respectively.

In the problem model, each workflow is represented as a *Directed Acyclic Graph* (DAG), i.e. a precedence task graph $q = (V, E)$, where the set of nodes V contains each task α $(1 \leq \alpha \leq m)$ of the parallel program q. The set of (directed) edges E represents the dependencies between tasks, a partial order $\alpha \prec \beta$ that models the precedence constraints: an edge $e_{\alpha\beta} \in E$ means that task β cannot start its execution before task α is completed. Communication costs between tasks are not considered because they occur always within the same datacenter.

We propose dealing with large workloads, so the largest problem instances are composed of thousands of workflows (this means hundreds of thousands of tasks) to be scheduled onto a federation of several datacenters comprised of hundreds to thousands of processing elements.

We model this with the multiobjective problem min (f_M, f_E, f_S), that proposes the *simultaneous* optimization of the *makespan* (f_M), the *energy consumption* (f_E), and the *violations of SLA* (f_S):

- The makespan evaluates the total time to execute a set of workflows, according to the expression in Eq. (4.1), where $\vec{x}$ represents an allocation, k is the number of available datacenters, and CT_r is the completion time of datacenter r (DC_r).

$$f_M(\vec{x}) = \max_{0 \leq r \leq k} CT_r \tag{4.1}$$

- The *energy consumption* function for a set of workflows executed in a certain datacenter is defined in Eq. (4.2), using the energy model for multi-core architectures by Nesmachnow et al. (2013), where f_1 is the higher-level scheduling function, and f_2 is the lower-level scheduling function. The total energy consumption takes into account both the energy required to execute the tasks assigned to each computing resource within a datacenter, and the energy that each resource consumes in idle state.

$$f_E(\vec{x}) = \sum_{r \in DC} \sum_{\substack{q \in Q: \\ f_1(q)=r}} \sum_{\substack{wt_\alpha \in WT_q: \\ f_2(wt_\alpha)=s_j}} \frac{o(wt_\alpha)}{ops(s_j)} \times e_{s_j}^{max} + \sum_{s_j \in S_r} e_{s_j}^{idle} \tag{4.2}$$

- The *violations of SLA agreements* is defined as the number of applications that violate the user's SLA agreement, i.e., the number of applications that do not finish before their deadline, over the allowed limit specified by the SLA (Eq. 4.3), where $Violated(q)$ takes value 1 when the deadline of workflow q is violated ($ftw_q > d_q$, being ftw_q the finishing time of workflow q, according to the schedule) and 0 otherwise, $SLA_{u_i} = \{0.9, 0.94, 0.98\}$ (for large problem instances) and $WF(u_i)$ is the number of workflows submitted by user u_i.

$$f_S(\vec{x}) = \sum_{u_i \in U} \max \left(0, \left[\sum_{q \in wu(u_i)} Violated(q) - (1 - SLA_{u_i}) \times WF(u_i)) \right] \right) \tag{4.3}$$

In this article, we study the optimization problem from the point of view of the computing system (i.e. the infrastructure administration), thus we use two system-related objectives in the problem formulation. Additionally, we consider a QoS-related objective such as the number of workflow deadlines satisfied violated, taking into account the point of view of the customer/user in the problem formulation.

4.2.2 Mathematical formulation

The mathematical formulation of the Energy-aware SLA Scheduling Problem in federated datacenters (E-SLA-SP) is presented next.

General elements and constants. The global elements in the problem definition include:

- m tasks, $\alpha = 1 \dots m$;
- n workflows, $q = 1 \dots n$;
- s machines, $j = 1 \dots s$;
- k datacenters, $r = 1 \dots k$;
- o users, $p = 1 \dots o$;
- u, the periods of time in the scheduling horizon, $i = 1 \dots u$;
- dcm_{rj}, equals 1 if machine j belong to datacenter r, 0 otherwise;
- c_j^{max}, the number of cores available in machine j;
- e_j^{idle}, the energy consumption per period of time of machine j when in idle state;
- e_j^{max}, the energy consumption per period of time of machine j when in processing state;
- d_q, the deadline before workflow q should be executed;
- $wt_{q\alpha}$, equals 1 if task α comprises workflow q, 0 otherwise;
- $p_{\alpha\hat{\alpha}}$ equals 1 if task α must precede task $\hat{\alpha}$, 0 otherwise;
- c_α, the number of cores required for executing task α;
- $len_{\alpha j}$, the number of time periods required to execute task α in machine j;
- wu_{qp}, equals 1 if workflow q was submitted by user p, 0 otherwise;
- sl_p, the maximum acceptable number of tasks with unsatisfied deadlines for user p.

Variables. Three decision variables are considered:

- w_{qr}: workflow q is assigned to be executed by datacenter r;
- $x_{\alpha j}$: task α is assigned to be executed by machine j;
- $y_{\alpha ji}$: task α is to be executed by machine j at time period i;

Four related variables are defined:

- st_α, the starting time period of task α;
- ft_α, the finishing time period of task α;
- ftw_q, the finishing time period of workflow q;
- v_q, equals 1 if workflow q violates its deadline.

Formulation. The mathematical formulation of the optimization problem is presented in Eq. (4.4).

The problem objectives are formulated first. Eq. (4.4a) formulates the *makespan* objective, $f_M(\vec{x})$. Eq. (4.4b) presents the mathematical formulation of the *energy consumption* objective, $f_E(\vec{x})$. Finally, Eq. (4.4c) formulates the *violations of SLA* objective, $f_S(\vec{x})$.

Regarding the problem constraints, Eq. (4.4d) states that each task must be included in one and only one workflow; Eq. (4.4e) indicates that each machine must comprise one and only one datacenter; Eq. (4.4f) states that each workflow must be assigned to one and only one datacenter; and Eq. (4.4g) indicates that each task must be assigned to one and only one machine.

Eq. (4.4h) guarantees that at any time period i, the number of cores required by all the tasks assigned to a machine j must be less or equal to the number of cores available in j. The constraint in Eq. (4.4i) asserts the execution time of task α must be zero on all machines to which it is not assigned. Eq. (4.4j) states st_α is the first time period in which α is executed, while Eq. (4.4k) states ft_α is the last time instance in which α is executed. Eq. (4.4l) indicates that all tasks preceding task $\hat{\alpha}$ must finish their executing before $\hat{\alpha}$ starts. Eq. (4.4m) indicates that all tasks of a given workflow must be assigned to the same datacenter. Eq. (4.4n) states the finishing time of workflow q is the latest finishing time of its tasks. Eq. (4.4p) and (4.4q) assert that v_q equals 1 when the deadline of workflow q is violated, and 0 otherwise. Finally, Eqs. (4.4r) are the integrality constraints for the variables used in the formulation.

$$\text{Minimize} \quad \max_{\alpha=1\ldots m} ft_\alpha \tag{4.4a}$$

$$\text{Minimize} \quad \sum_{j=1}^{s} \left(\max_{\alpha=1\ldots m} x_{\alpha j} \times ft_\alpha \times e_j^{idle} \right) + \sum_{\alpha=1}^{m} \left(x_{\alpha j} \times len_{\alpha j} \times \left(e_j^{max} - e_j^{idle} \right) \right) \tag{4.4b}$$

$$\text{Minimize} \quad \sum_{p=1}^{o} \max \left\{ 0, \left(\sum_{q=1}^{n} wu_{qp} \times v_q \right) - sl_p \right\} \tag{4.4c}$$

subject to

$$\sum_{\alpha=1}^{m} wt_{q\alpha} = 1 \quad (q = 1 \ldots n) \tag{4.4d}$$

$$\sum_{j=1}^{s} dcm_{rj} = 1 \quad (r = 1 \ldots k) \tag{4.4e}$$

$$\sum_{r=1}^{k} w_{qr} = 1 \quad (q = 1 \ldots n) \tag{4.4f}$$

$$\sum_{j=1}^{s} x_{\alpha j} = 1 \quad (\alpha = 1 \ldots m) \tag{4.4g}$$

$$\sum_{\alpha=1}^{m} y_{\alpha j i} \times c_\alpha \leq c_j^{max} \quad (i = 1 \ldots u; j = 1 \ldots s) \tag{4.4h}$$

$$\sum_{i=1}^{u} y_{\alpha j i} = x_{\alpha j} \times len_{\alpha j} \quad (\alpha = 1 \ldots m; j = 1 \ldots s) \tag{4.4i}$$

$$\sum_{i=1}^{st_\alpha - 1} \sum_{j=1}^{s} y_{\alpha j i} = 0 \quad (\alpha = 1 \ldots m) \tag{4.4j}$$

$$\sum_{i=ft_\alpha+1}^{u} \sum_{j=1}^{s} y_{\alpha j i} = 0 \quad (\alpha = 1 \ldots m) \tag{4.4k}$$

$$p_{\alpha\hat{\alpha}} \times ft_\alpha \leq p_{\alpha\hat{\alpha}} \times st_{\hat{\alpha}} \quad (\alpha = 1 \ldots m; \hat{\alpha} = 1 \ldots m) \tag{4.4l}$$

$$\sum_{\alpha=1}^{m} \sum_{j=1}^{s} wt_{q\alpha} \times dcm_{rj} \times x_{\alpha j} = w_{qr} \times \sum_{\alpha=1}^{m} wt_{q\alpha} \quad (q = 1 \ldots n; r = 1 \ldots k) \tag{4.4m}$$

$$ftw_q = \max_{\alpha=1\ldots m} wt_{q\alpha} \times ft_\alpha \quad (q = 1 \ldots n) \tag{4.4n}$$

$$\tag{4.4o}$$

$$ftw_q \times (1 - v_q) \leq d_q \quad (q = 1 \ldots n) \tag{4.4p}$$

$$ftw_q > d_q \times v_q \quad (q = 1 \ldots n) \tag{4.4q}$$

$$w_{qr}, x_{\alpha j}, y_{\alpha ji}, v_q \in \{0, 1\},\ st_\alpha, ft_\alpha, ftw_q \geq 0$$
$$(q = 1 \ldots n; r = 1 \ldots k; \alpha = 1 \ldots m; j = 1 \ldots s; i = 1 \ldots u) \tag{4.4r}$$

4.2.3 Related work

There is a large number of works dealing with energy-aware scheduling in computing systems that recently appeared in the literature. Mainly, we can classify them as *independent* and *simultaneous* scheduling, according to the way in which they deal with the optimization objectives.

In the *independent* approach, energy and performance are optimized as separate goals, not taking into account their relationships explicitly. Therefore, this kind of problems can be solved using existing scheduling algorithms that optimize classic performance metrics (i.e., makespan, flowtime, etc.) combined with a slack reclamation technique to deal with energy optimization, such as dynamic voltage scaling (DVS)/dynamic voltage and frequency scaling (DVFS) (Baskiyar and Abdel-Kader, 2010; Rizvandi et al., 2011). This approach is limited in the sense that the optimization of one of the objectives is always restricted by the optimization process performed for the other one.

In the *simultaneous* approach, performance and energy are optimized at the same time, therefore the scheduling is modeled as a multi-constrained, bi-objective optimization problem. The algorithms are oriented to find Pareto optimal schedules; i.e., where no scheduling decision can strictly dominate the other ones with better performance and lower energy consumption at the same time. This approach provides a more holistic and realistic problem solving technique for energy-aware scheduling in datacenters. We follow the simultaneous approach in this work, thus we briefly review the main related works about simultaneous optimization of energy and performance metrics next.

We can find in the literature a number of works that focus on finding only one trade-off solution instead of a Pareto set. Following this approach, Khan and Ahmad (2009) applied the concept of Nash Bargaining Solution from cooperative game theory to find optimal trade-off schedules for workflows of independent tasks, simultaneously minimizing makespan and energy on a DVS-enabled grid system.

Another relevant work is the one done by Lee and Zomaya (2011), where several DVS-based heuristics to minimize the weighted sum of makespan and energy are studied. In order to escape from local optima, authors implement a conservative local search technique that allows slight modifications on the schedules to enhance makespan if it does not imply increasing the energy consumption. Later, Mezmaz et al. (2011) improved the previous work by proposing a parallel bi-objective hybrid genetic algorithm (GA) for the same problem, using the cooperative island/multi-start farmer-worker model, significantly reducing the execution time of the scheduling method, and reporting a Pareto front of non-dominated solutions as a result. Pecero et al. (2011) proposed a two-phase bi-objective algorithm based on the Greedy Randomized Adaptive Search Procedure (GRASP) that applies a DVS-aware bi-objective local search to generate a set of Pareto solutions.

Li et al. (2009) introduced a MinMin-based online dynamic power management strategy with multiple power-saving states to reduce energy consumption of scheduling algorithms. Pinel et al. (2013) proposed a double minimization approach for scheduling independent tasks on grids with energy considerations, first applying a MinMin approach to optimize the makespan, and then a local search to minimize energy consumption. Lindberg et al. (2012) proposed six greedy algorithms and two GAs for solving the makespan-energy scheduling problem subject to deadline and memory requirements.

In our previous work (Nesmachnow et al., 2013), we proposed a new energy consumption model for multi-core computing systems, not based on DVS or other specific techniques for power/energy management. Instead, we focus on the energy required: *i*) to execute tasks at full capacity, *ii*) when the multi-core machines are partially used, and *iii*) to keep machines in idle state. We proposed twenty fast list scheduling methods adapted to solve the bi-objective problem of executing tasks on a single cluster node with the goals of minimizing the time and energy required. Iturriaga et al. (2013) studied the same problem with a parallel multiobjective local search based on Pareto dominance, showing its superior performance versus the previously mentioned deterministic heuristics. These works deal with the problem of scheduling independent Bag-of-Tasks (BoT) applications.

In our recent work (Dorronsoro et al., 2014a), we considered the multiobjective problem of scheduling large workflows of applications (modeled as DAGs) into a number of clusters. The objectives to optimize were the makespan of

the schedule, the energy consumption, and a Quality of Service (QoS) metric of the solutions, computed by three cost functions that depend on the time exceeded to complete jobs, with respect to their deadline. The problem was solved with a two-levels approach, using deterministic schedules in every level.

In the research reported in the present article we define a more difficult, interesting, and realistic problem variant with respect to the one presented in our previous work (Dorronsoro et al., 2014a). Here, we consider the problem of scheduling large workflows of parallel applications on a federated cloud infrastructure, composed by a number of datacenters that are geographically distributed. In addition, we take into account the Service Level Agreement (SLAs) signed between the provider and its customers. These SLAs must be respected by the provider, whenever possible. Therefore, our problem optimizes the makespan and energy consumption of the solutions, as well as the number of SLA agreements respected by the provider. The novelty of this work in the context of federation of distributed datacenters is that we consider QoS as an optimization objective, and provide lower-bounds for the online heuristics considering medium- and large-sized problem instances with Constraint Programming and offline MOEAs.

4.3 Methodology and techniques

This section describes the methodology and techniques applied for solving the scheduling problem.

4.3.1 List scheduling heuristics

List scheduling heuristics are deterministic static scheduling methods that work by assigning priorities to tasks based on a particular ad-hoc criteria. After that, the list of tasks is sorted by decreasing priority and each task is assigned to a processor, regarding the task priority and the processor availability (Ibarra and Kim, 1977). Algorithm 4 presents the general schema of a list scheduling method.

List scheduling heuristics have been applied to minimize the makespan metric in classic works on the scheduling field (Braun et al., 2001). We have extended this concept to multiobjective scheduling of independent tasks considering makespan and energy, in our previous work (Nesmachnow et al., 2013).

Algorithm 4 Schema of a list scheduling algorithm

```
1: while tasks left to assign do
2:     determine the most suitable task according to the chosen criterion
3:     for each task to assign, each machine do
4:         evaluate criterion (task, machine)
5:     end for
6:     assign the selected task to the selected machine
7: end while
8: return task assignment
```

In the research reported in the present article, we apply a number of scheduling heuristics specifically adapted for the two-level energy-aware scheduling problem in a federation of datacenters (see the details in Section 4.4.1).

4.3.2 Multiobjective evolutionary algorithms

Evolutionary Algorithms (EAs) are non-deterministic metaheuristic methods that emulate the evolution of species in nature to solve optimization, search, and learning problems (Bäck et al., 1997). In the last thirty years, EAs have been successfully applied for solving many high-complexity optimization problems (Chiong et al., 2011; Nesmachnow, 2014).

MultiObjective Evolutionary Algorithms (MOEAs) (Coello Coello et al., 2007; Deb, 2001) are specific evolutionary techniques designed to solve multiobjective optimization problems. They have been applied to solve hard optimization problems, obtaining accurate results when solving real-life problems in many research areas. Unlike many traditional methods for multiobjective optimization, MOEAs are able to find a set with several solutions in a single execution, since they work with a population of tentative solutions. MOEAs must be designed taking into account two goals at the same time: *i*) approximating the Pareto front, usually applying a Pareto-based evolutionary search, and *ii*) maintaining diversity instead of converging to a reduced section of the Pareto front, usually accomplished by using specific techniques also used in multimodal function optimization (sharing, crowding, etc.).

In this work, we apply two MOEAs to solve the E-SLA-SP optimization problem:

- *Non-dominated Sorting Genetic Algorithm, version II* (NSGA-II) (Deb, 2001). NSGA-II is a state-of-the-art MOEA that has been successfully applied to solve optimization problems in many application areas.

NSGA-II includes features to deal with three criticized issues on its predecessor NSGA, in order to improve the evolutionary search: *i*) an improved non-dominated elitist ordering that diminishes the complexity of the dominance check; *ii*) a crowding technique for diversity preservation; and *iii*) a new fitness assignment method that considers the crowding distance values.

- *Multiobjective Cellular Genetic Algorithm* (MOCell) (Nebro et al., 2009). MOCell is characterized by the use of a decentralized (*cellular*) population and an archive to store the non-dominated solutions found during the search. In MOCell, individuals are arranged into a two dimensional toroidal grid, and only those individuals that are close to each other in the mesh are allowed to interact. The main advantage of using this population structure, called *cellular*, is that individuals are isolated by distance, and therefore good solutions will spread slowly through the population mesh, consequently keeping the diversity of solutions for longer (Alba and Dorronsoro, 2008). From all the studied versions of MOCell reported by Nebro et al. (2009), we use here the so-called *aMOCell4*, which takes advantage of the non-dominated solutions stored in the archive by choosing one of them as one of the parents during the breeding loop.

Next section presents the main characteristics and implementation details of the proposed MOEAs algorithm to solve the E-SLA-SP.

4.3.3 Lower bounds for the problem

Specific lower bounds are computed for each of the problem objectives, by solving a relaxation of the general scheduling problem. The relaxed problem formulation considers most of the original problem variables and constraints, but no restrictions are imposed over the computational resources where to execute a task (i.e., all machines are considered as they are in a single large pool of resources in the same datacenter). This assumption allows us to discard the decision variable that associates workflows to datacenters (w_{qr}), the binary variable that indicates if a given machine belongs to a certain datacenter (dcm_{rj}), as well as the constraints related to the geographical location of both resources and workflows, formulated in Eq. (4.4e), (4.4f), and (4.4m).

For computing the lower bounds, we built a Constraint Programming (CP) model using IBM CPLEX Optimizer version 12.5. The CP model is roughly

based on the mathematical model presented in Section 4.2.2, but taking into account the previously described relaxation. In addition, the CP model takes advantage of special scheduling constructs provided by the CPLEX engine, such as the variables `interval` and `sequence`. In the CPLEX engine, an `interval` variable represents a lapse of time described by a start and an end value, while a `sequence` variable represents a total order over a set of `interval` variables. In the proposed model, `interval` variables are used to model tasks, and `sequence` variables are used to model the execution queue of each machine core.

Listing 4.1 presents the schema of the CP model for the lower bound computation on the makespan objective. The CP models for calculating the lower bounds for the energy consumption and SLA violation objectives follow the same general schema, but changing the objective function computation in the minimization (line 18 in Listing 4.1).

The exact resolution method applied by the CP model is only viable to execute in reasonable times for small-sized problem instances. Taking this argument into consideration, a stopping condition of 14 hours was applied to finish the execution of the CP model; if the optimal solution is not found before the stopping condition is met, then the current best solution found is returned. This time stopping criterion allows optimally solving the relaxed problem version for all type of considered instances, except for the Single-Task class, which involves a significantly larger number of tasks than the other workflow types (see Section 4.5.1 for a description of the instance types solved in the experimental analysis).

4.4 The proposed algorithms

This section presents the algorithms used in this work to solve the energy-SLA scheduling problem in a federation of distributed datacenters.

Listing 4.1: Schema of the CP model for lower bound computation for the makespan objective

```
nbTaskCores = ...; //Load workloads data from file
range TaskCores = 1..nbTaskCores;
nbTasks = ...;
range Tasks = 1..nbTasks;
{int} tasks[Tasks] = ...;
{int} preccs[TaskCores] = ...; //Load tasks dependencies
duration[TaskCores,MachineCores] = ...;
nbMachineCores = ...; //Load datacenters scenario
range MachineCores = 1..nbMachineCores;
int nbMachines = ...;
range Machines = 1..nbMachines;
{int} machines[Machines] = ...;
int nbDC = ...;
range DC = 1..nbDC;
{int} dc[DC] = ...;
dvar interval task[t in TaskCores];
dvar interval opttask[t in TaskCores][m in MachineCores]
  optional size duration[t][m];
dvar sequence tool[m in MachineCores] in all(t in TaskCores)
  opttask[t][m];
//Minimize the makespan
minimize max(t in TaskCores) endOf(task[t]);
subject to {
  forall(t in TaskCores)
    //Each job needs one unary resource of the
    //alternative set opttask
    alternative(task[t], all(m in Machines) opttask[t][m]);
  forall(tg in Tasks) {
    //All cores of the task start and end at the same time
    synchronize(task[first(tasks[tg])],
      all(tg2 in tasks[tg]) task[tg2]);
    //All cores of the task are assigned to the same machine
    (sum(mg in Machines)
      ((sum(t in tasks[tg], m in machines[mg])
        presenceOf(opttask[t][m]))==card(tasks[tg]))) == 1;
  }
  forall(j in Jobs) {
    (sum(c in DC)
      (card(jobTasks[j])==(sum(t in jobTasks[j], m in dc[c])
        presenceOf(opttask[t][m])))) == 1;
  }
  forall(m in MachineCores)
    noOverlap(tool[m]); //No overlap on machines
  //Precedence constraints
  forall (t1 in TaskCores)
    forall (t2 in preccs[t1])
      endBeforeStart(task[t2], task[t1]);
};
```

4.4.1 Heuristics

The heuristic schedulers we use in this work are based on those presented in our previous work (Dorronsoro et al., 2014a), but adapted to the problem of multiobjective scheduling in a federation of datacenters. The heuristics are

two-levels schedulers: *i*) in the first level, the front-end heuristic assigns workflows to the different datacenters, specifying the order in which they should be locally scheduled; *ii*) in the second level, there is a multi-workflow scheduler applying the backfilling strategy, based on the well-known HEFT heuristic.

For the first level, we study here specific variants of the four heuristics analyzed by Dorronsoro et al. (2014a), namely round robin, load balancing, and two versions of MaxMin heuristic: the standard one plus a multiobjective version that considers not only execution time but energy consumption too. We also propose a new one, based on MinMin, that takes decisions based on both computation time and energy consumption.

A short description of the high level heuristics is given next:

1. *Round Robin* (RR): This classic approach iteratively assigns jobs to the next available datacenter. Should the selected datacenter cannot satisfy core requirements for all tasks of a job, the datacenter will be skipped; this iterative procedure is continued until each job is assigned.
2. *Load Balance* (LB): This heuristic assigns jobs to balance the workloads of the datacenters. To this end, jobs are first sorted according to their core requirement; i.e., the maximum number of cores required by any task of the job. Jobs with higher cores requirements are then prioritized for assignment. Based on their order, the LB scheduler dispatches each job to the datacenter with lowest number of assigned jobs that can execute the job.
3. *MaxMin*: In this heuristic, each job is assigned to the datacenters that can execute it faster. To this end, in each iteration of this heuristic, the job that leads to the overall maximum completion time is selected and assigned to a datacenter –considering its core requirement– that can finish it earlier; this procedure continues until all jobs are assigned. As a result of this scheduling policy, longer jobs that require more execution time are scheduled before shorter jobs that have less effect on the overall makespan of a system.
4. *MaxMIN*: Similar to MaxMin, this heuristic assigns jobs to datacenters considering their already assigned jobs, however to minimize their overall energy consumption; i.e., in MaxMIN, jobs with larger energy consumption are selected and assigned to datacenters that can execute them earlier considering their core requirements. Unlike MaxMin that

only optimize the execution time of jobs and hopes to balance their energy consumption through load balancing, MaxMIN actively selects the jobs with larger energy consumption and prioritize their allocation with the hope of optimizing both objectives.

5. *MinMIN*: Similar to MaxMIN, but in this case in each iteration of the heuristic, the job that leads to the overall minimum completion time (instead of the maximum) is selected and assigned to a datacenter – considering its core requirement– that can finish it with the lowest energy consumption.

We chose the best low-level (i.e., operating at the datacenter level) scheduler from our previous work, namely EFTH (Earliest Finishing Time Hole). It schedules the workflows in the order established by the first level scheduler. The schedule of the first workflow is performed exactly as HEFT would do. The next workflows are scheduled in order, one by one, following the same strategy of HEFT, but using a backfilling strategy: EFTH tries to fill all gaps (or holes) in which the processors are idle due to dependencies between tasks. In this sense, EFTH assigns a task to the whole that can finish it the earliest.

We use the proposed heuristics as baseline schedulers in order to compare the results computed by the proposed MOEAs. The heuristics represent a typical scenario for datacenter operation, using a load balancing/maximizing utilization method, such as the ones traditionally used in current cluster, grid, and cloud management systems. In addition, for medium-sized problem instances, we compare the MOEA results with the lower bounds computed using the relaxation techniques described in Section 4.3.3.

4.4.2 Multiobjective Evolutionary Algorithms

The MOEAs used in this study are NSGA-II and MOCell. The main features of each algorithm as generic optimization techniques were already described in Section 4.3.2. This section presents the details of the implemented versions of NSGA-II and MOCell to solve the scheduling problem considered in this article.

Solution encoding. Solutions are encoded as a permutation of integer numbers with length $n+k-1$ (being n the number of workflows and k the number of datacenters). This way, numbers 0 to $n-1$ represent the n workflows and

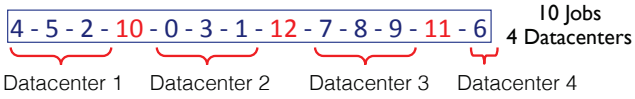


Figure 4.2: Solution encoding.

numbers n to $n + k - 1$ are splitters, used to separate workflows assigned to the different datacenters. A graphical representation of the solution encoding can be found in Figure 4.2. We can see that those workflows that appear until the first splitter in the solution are assigned to datacenter 1, the following workflows until the next splitter are mapped into datacenter 2, and so on. Moving one workflow from one datacenter to another is as simple as changing its identificator to the corresponding place in the permutation. Not assigning any workflow to a datacenter is possible just by putting two splitters in consecutive positions.

The use of this permutation representation allows the genetic algorithm to decide not only the workflows to assign to every datacenter, but also the order in which they must be processed by the local scheduler (this important issue is not considered by the heuristics, for which this is defined according to the arrival order of workflows in the federated datacenter). Additionally, it allows us the use of well known generic operators for permutation representations.

Fitness values. The three objective functions defined in Section 4.2.1 for the optimization problem (i.e., makespan, energy consumption, and SLA violations) are taken into account to define a fitness value for each candidate solution. Then, the quality of each solution is evaluated accordingly to the fitness assignment schema proposed in each MOEA. NSGA-II applies the fitness ranking based on non-domination sorting concept (Deb, 2001), while MOCell uses the Strength Raw Fitness technique (SRF) (Zitzler et al., 2001).

Population initialization. The initial population is created randomly using an uniform distribution function to create random permutations representing the initial tentative solutions to the problem.

Selection operator. Selection is performed using the binary tournament method, considering Pareto dominance and crowding distance in the space of the objective functions. This method randomly selects two solutions from the population. If one of the selected solutions is dominated, then

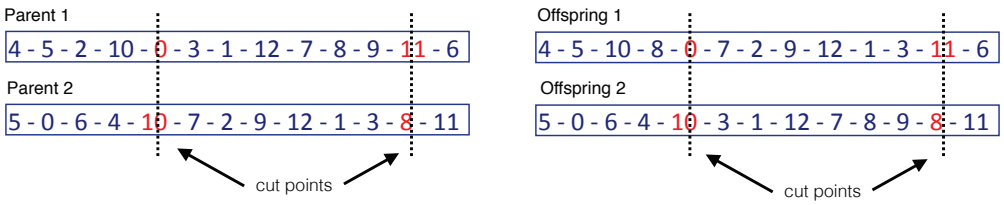


Figure 4.3: The PMX recombination operator.

it is discarded and the non-dominated solution is selected. If both solutions are non-dominated, then the solution which is in the most crowded region is discarded and the remaining solution is selected.

Crossover operator. The well-known Partially Matched Crossover (PMX) method is used as the crossover operator. Its behavior is graphically represented in Figure 4.3. In this operator, two positions are randomly selected from the solution as cutting points. All jobs in between these two points are swapped. The remaining jobs are rearranged using position wise exchanges, maintaining its original ordering information.

Mutation operator. A simple exchange method is used as the mutation operator. This method works by randomly selecting two jobs and swapping their positions.

Repair operator. This special operator repairs a non-feasible solution turning it into a feasible one. It is applied right after the crossover and mutation operators in order to repair any non-feasible feature introduced by these operators. The operator checks, for every workflow, whether it can be executed in the assigned datacenter or not. In the latter case, it is reassigned to the next datacenter that can execute it. A workflow can be executed in a datacenter if the number of cores of its servers is not less than the maximum number of cores required by any task in the workflow.

External file management (MOCell). We modified the original MOCell by using the SRF technique to manage the archive of non-dominated solutions (therefore, the resulting algorithm is called MOCellSRF). The reason for this change is that SRF generally provides fronts with better diversity for three dimensional problems (Dorronsoro et al., 2014b), as the one considered in this work.

Table 4.1: Parameter configuration of the proposed MOEAs.

Subpopulation size	100
Archive (MOCellSRF)	
size	100
management	Strength Raw Fitness
Max. evaluations	$25,000$
Pop. initialization	Random
Neighborhood (MOCellSRF)	C9
Selection	Binary tournament
Recombination	PMX
Probability	0.9
Mutation	Swap
Probability	$1/number\_of\_variables$
Independent runs	30

Parameter configuration. The configuration of the algorithms is detailed in Table 4.1, based on a preliminary parameterization study of both MOEAs. The population size and the external archive size (for MOCellSRF) are set to 100 solutions. As mentioned before, the Strength Raw Fitness method is used to manage the archive of non-dominated solutions in MOCellSRF. The population is randomly initialized, and evolved until $25,000$ schedule evaluations are performed, since neither MOEA evolved considerably after 25,000 evaluations. Parents are selected by binary tournament from the whole population in NSGA-II or from the eight surrounding neighbors in the case of MOCellSRF. The recombination and mutation operators are PMX and Swap, as explained before, and the parametrization study showed that the best results are computed when using probabilities of 0.9 and $1/number\_of\_variables$, respectively. Finally, in the experimental analysis we perform 30 independent runs of the algorithms for every problem instance.

4.5 Experimental evaluation

This section reports the experimental evaluation of the proposed MOEAs for energy/SLA scheduling in a federation of distributed datacenters.

Table 4.2: Characteristics of the processors considered for the DC infrastructures

processor	*frequency*	*cores*	*GFLOPS*	E_{IDLE}	E_{MAX}	*GFLOPS/core*
Intel Celeron 430	1.80 GHz	1	7.20	75.0W	94.0W	7.20
Intel Pentium E5300	2.60 GHz	2	20.80	68.0W	109.0W	10.40
Intel Core i7 870	2.93 GHz	4	46.88	76.0W	214.0W	11.72
Intel Core i5 661	3.33 GHz	2	26.64	74.0W	131.0W	13.32
Intel Core i7 980 XE	3.33 GHz	6	107.60	102.0W	210.0W	17.93

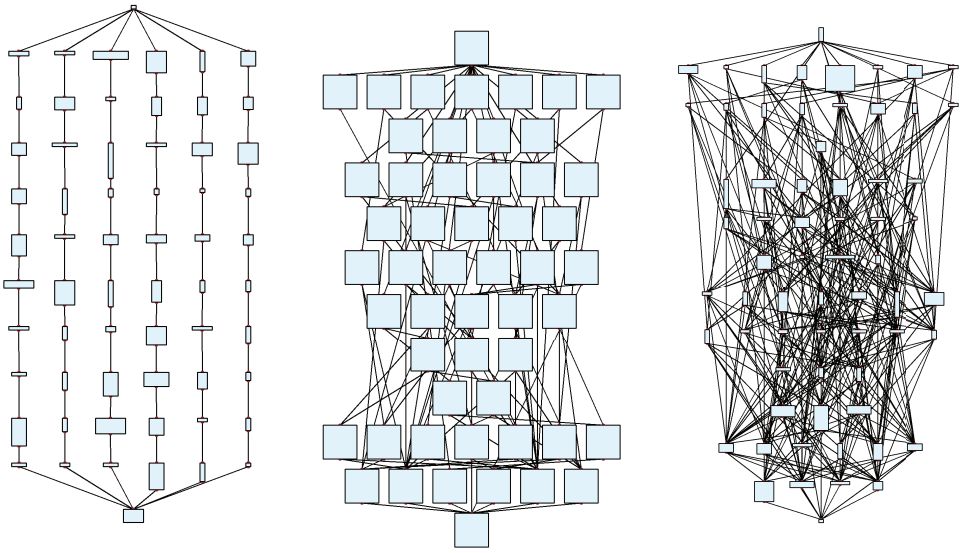
4.5.1 Problem instances

Problem instances are defined by a *workload*, a *computational scenario* and a *list of Service Level Agreement* rates.

Regarding workloads, we use five different models to consider different user applications submitted to the system: (a) *Series-Parallel* (b) *Heterogeneous-Parallel*, (c) *Homogeneous-Parallel*, (d) *Single-Task*, and (e) *Mix*. The Series-Parallel model represents jobs that can be split into concurrent threads/processes running in parallel. Heterogeneous-Parallel represent a generic job composed of non-identical computational tasks with arbitrary precedences. Homogeneous-Parallel represents jobs composed of identical computational blocks. Single-Task represents jobs of only one task. Finally, the Mix category combines workflows of all other categories, i.e., Series-Parallel (30% of all workflows), Heterogeneous-Parallel (30%), Homogeneous-Parallel (30%), and Single-Task (10%) jobs. Fig. 4.4 shows the overall shape of difference workflow types aimed to reflect real high performance computing applications. Here, each block represents a computational task with specific characteristics; i.e., execution time (height of a block) and number of processors (width of a block). All workflows used in the experimental analysis were generated using the SchMng application (Taheri et al., 2013).

Regarding the computational infrastructure, the federation of datacenters combines nowadays Intel processor servers with one to six cores, listed in Table 4.2. Each datacenter is comprised of one type of processor, while different datacenters are comprised of different type of processors.

Medium-size problem instances. A benchmark set of 9 medium-sized workflow batches was generated for comparing the efficacy of proposed MOEA and scheduling heuristics with an exact method based on Constraint Programming (CP). The workflow batches were build with a total of 600 tasks each, hence



(a) Series-Parallel **(b)** Homogeneous-Parallel **(c)** Heterogeneous-Parallel

(d) Single-Task

Figure 4.4: Workflow types used in the experimental analysis

the number of jobs in each workflow batch ranges from 10 up to 250. We considered a fixed scenario with a federation of two datacenters with a total of 80 cores. Such medium-sized problem instances are—nevertheless—realistic and usually appear in the related literature (Goiri et al., 2013, 2014; Jayasinghe et al., 2011). For the SLA levels, we assigned all the jobs equitably between three different users, and considered three values for the number of tasks allowed to miss their deadline, one for each user: {1, 2, 3}.

Large problem instances. A benchmark set of 125 different workflow batches was generated for the experimental evaluation of the proposed heuristics and MOEAs over large problem instances. The number of tasks in workflows ranges from 3 to 132, in addition to the Single-Task ones, composed of only one task.

In the benchmark set of 125 batch of workflows, 25 correspond to $1,000$ Series-Parallel workflows ($25,000$ workflows altogether), 25 are composed of $1,000$ Heterogeneous-Parallel workflows ($25,000$ workflows altogether), 25 more are $1,000$ Homogeneous-Parallel workflows ($25,000$ workflows altogether), 25 other batches are single-task jobs, and the remaining 25 are a mix of them, including a combination of different workflow types (300 Heterogeneous-Parallel workflows, 300 Homogeneous-Parallel workflows, 300 Series-Parallel workflows, and 100 Single-Task applications). A total number of $\mathbf{125,000}$

workflows are studied in the experimental analysis. Regarding the computational infrastructure, we consider scenarios involving a federation of five datacenters, with up to 100 processors each, from the ones defined in Table 4.2. For the SLA levels, we define three values for the number of tasks to be completed on time: {90%, 94%, 98%}. These are realistic values for QoS offered by current datacenter and cloud computing facilities, similar to QoS values considered in the related literature (de Assuncao et al., 2009; Kim et al., 2007; Moon et al., 2011).

The benchmark set of workflows, scenarios, and SLA levels is publicly available, it can be accessed/downloaded by contacting the authors.

4.5.2 Experimental setup

We detail in this section the procedures we followed during our experiments. In case of the deterministic algorithms studied (both the exact and heuristic algorithms) we report the average values obtained for all 25 instances of each kind. In the case of the non-deterministic ones (i.e., the MOEAs), we performed 30 independent runs of the algorithms for every problem instance and compare the algorithms according to the obtained results of every independent runs for each problem instance. Therefore the comparisons are done on 750 different results for each problem kind. After the mentioned experiments, the following metrics are applied to the obtained Pareto fronts in order to quantify their quality in terms of different features as accuracy and diversity of solutions:

- Hypervolume (HV) (Zitzler and Thiele, 1999): calculates the m-dimensional volume (in the objective space) covered by the solutions in the evaluated Pareto front Q and a dominated reference point W. Mathematically, for each solution $i \in Q$, a hypercube v_i is constructed with the reference point W (e.g., constructed with a vector of worst objective function values) and the solution i as the diagonal corners of the hypercube. Thereafter, a union of all hypercubes is found and its hypervolume is calculated, as shown in Eq. (4.5). Algorithms with the highest HV value perform best as this metric takes its maximum value when all the solutions in the evaluated Pareto front belong to the optimal one.

$$HV = \text{volume}\left(\bigcup_{i=1}^{|Q|} v_i\right) \tag{4.5}$$

- Spread (Δ) (Deb et al., 2002): this indicator measures the extent of spread by the set of computed solutions. It is defined by Eq. (4.6), where d_i is the Euclidean distance between consecutive solutions, $\bar{d}$ is the mean of these distances, and d_f and d_l are the Euclidean distances to the *extreme* solutions of the optimal Pareto front in the objective space. This indicator takes a zero value for an ideal distribution, pointing out a perfect spread of the solutions in the Pareto front.

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (N-1)\bar{d}} \tag{4.6}$$

- Unary Additive Epsilon ($I^1_{\varepsilon+}$) (Knowles et al., 2006): this indicator provides a measure of the convergence, i.e. of the distance to the optimal Pareto front. Given an approximation set of a problem, S, the $I^1_{\varepsilon+}$ indicator is a measure of the smallest distance needed to translate every point in S so that it dominates the true Pareto front of the problem S^*. More formally, given $\vec{e}^1 = (e^1_1, ..., e^1_m)$ and $\vec{e}^2 = (e^2_1, ..., e^2_m)$, where m is the number of objectives, and $\vec{e}^1 \prec_\varepsilon \vec{e}^2$ if and only if $\forall 1 \leq i \leq m : e^1_i < \varepsilon + e^2_i$, as defined by Eq. (4.7).

$$I^1_{\varepsilon+}(S) = \inf_{\varepsilon \in \mathbb{R}} \{ \forall \vec{e}^2 \in S^* \exists \vec{e}^1 \in S : \vec{e}^1 \prec_\varepsilon \vec{e}^2 \} \tag{4.7}$$

The previous metrics require the optimal Pareto front in order to compute accuracy of solutions and/or normalize the results (in order to avoid any bias due to different orders of magnitude in the objectives). Because the optimal Pareto front is not known for the considered problem, we use a *reference* Pareto front instead, as suggested by Dorronsoro et al. (2014b). This front is built by collecting the best non-dominated solutions found in all independent runs.

In order to assess statistical confidence to our conclusions, we perform the Wilcoxon signed rank test (Siegel, 1956), the Friedman test (Siegel, 1956) and the Holm-Bonferroni test (Holm, 1979) to check whether the results of the two compared algorithms follow the same distribution or not, with a 95% confidence level (i.e., if p-value ≤ 0.05 we say there is statistical difference between the compared distributions).

4.5.3 Development and execution platform

The heuristics are implemented in C, using the standard `stdlib` library and the `gcc` compiler. Regarding the two MOEAs, NSGA-II and MOCell, we use the implementation provided in jMetal framework (Durillo and Nebro, 2011), implemented in Java.

The experimental analysis was performed in cluster with Bull B505 servers, each one with six-core Intel Xeon CPU L5640 processor at 2.27GHz, 24 GB RAM, using a CentOS Linux operative system. The experimental platform is hosted as part of the HPC facility of the University of Luxembourg (platform website: `https://hpc.uni.lu`).

4.5.4 Numerical results

This subsection reports the numerical results of the proposed methods to solve the E-SLA-SP.

Medium-size problem instances

In this subsection we study the results computed when solving the medium-size problem instances. We compare the results computed by the heuristics and the MOEA with the lower bounds computed by the CP implementation. Regarding the MOEA, a total of 30 independent executions were performed for each medium-size problem instance. For the comparison we consider the average of the best makespan result computed in each execution.

In order to compare the computed results we introduce the lower bound gap metric, defined by Eq. (4.8), where *result* is the objective value computed by the resolution method evaluated (heuristics or MOEAs), and LB is the lower bound computed by the constraint programming method.

$$GAP = \frac{result - LB}{LB} \tag{4.8}$$

Table 4.3 presents the average and standard deviation gaps—indicated by a $\pm$ sign—to the CP lower bounds. It shows the makespan (GAP_M), energy consumption (GAP_E) and violations of SLA (GAP_S) gaps of the schedules computed by the scheduling heuristics and the proposed MOEAs, for medium-size problem instances. The best average results for every instance class are emphasized in **bold font**.

Table 4.3: Gaps (with respect to the CP lower bound) computed by the scheduling heuristics and the proposed MOEAs, for medium-size problem instances

Single-Task			
Algorithm	GAP_M	GAP_E	GAP_S
RR-EFT	17.8±15.1%	26.4±10.4%	**0.0±0.0%**
LB-EFT	20.7±15.5%	26.1±12.9%	**0.0±0.0%**
MaxMin-EFT	1.3±0.9%	29.8±12.9%	**0.0±0.0%**
MaxMIN-EFT	1.3±5.0%	29.8±12.9%	**0.0±0.0%**
MinMIN-EFT	45.8±17.0%	49.0±17.8%	**0.0±0.0%**
NSGA-II	1.4±1.3%	7.7±5.1%	**0.0±0.0%**
MOCellSRF	1.3±1.2%	8.8±5.7%	**0.0±0.0%**
Series-Parallel			
Algorithm	GAP_M	GAP_E	GAP_S
RR-EFT	5.6±8.5%	53.0±17.5%	**0.0±0.0%**
LB-EFT	5.6±8.5%	53.5±18.2%	**0.0±0.0%**
MaxMin-EFT	5.0±8.6%	51.6±18.4%	**0.0±0.0%**
MaxMIN-EFT	5.0±8.6%	51.6±18.4%	**0.0±0.0%**
MinMIN-EFT	9.1±19.0%	56.5±17.9%	**0.0±0.0%**
NSGA-II	5.0±8.6%	51.6±18.4%	**0.0±0.0%**
MOCellSRF	5.0±8.6%	51.6±18.4%	**0.0±0.0%**
Homogeneous-Parallel			
Algorithm	GAP_M	GAP_E	GAP_S
RR-EFT	5.2±5.0%	76.8±62.2%	**0.0±0.0%**
LB-EFT	5.2±5.0%	77.2±61.9%	**0.0±0.0%**
MaxMin-EFT	3.6±2.1%	78.2±61.5%	**0.0±0.0%**
MaxMIN-EFT	3.6±2.1%	78.2±61.5%	**0.0±0.0%**
MinMIN-EFT	5.2±5.0%	76.9±61.6%	**0.0±0.0%**
NSGA-II	5.2±5.0%	76.2±61.8%	**0.0±0.0%**
MOCellSRF	3.6±2.1%	76.2±61.8%	**0.0±0.0%**
Heterogeneous-Parallel			
Algorithm	GAP_M	GAP_E	GAP_S
RR-EFT	3.1±2.0%	70.8±39.5%	**0.0±0.0%**
LB-EFT	3.1±2.0%	68.2±40.6%	**0.0±0.0%**
MaxMin-EFT	3.1±2.0%	68.2±40.6%	**0.0±0.0%**
MaxMIN-EFT	3.1±2.0%	68.2±40.6%	**0.0±0.0%**
MinMIN-EFT	2.4±0.8%	65.8±38.0%	**0.0±0.0%**
NSGA-II	2.3±0.7%	63.4±38.9%	**0.0±0.0%**
MOCellSRF	2.3±0.7%	63.4±38.9%	**0.0±0.0%**
Mix			
Algorithm	GAP_M	GAP_E	GAP_S
RR-EFT	3.3±2.2%	72.9±45.8%	**0.0±0.0%**
LB-EFT	4.4±3.6%	74.5±45.9%	**0.0±0.0%**
MaxMin-EFT	3.9±3.6%	73.6±45.1%	**0.0±0.0%**
MaxMIN-EFT	3.9±3.6%	73.6±45.1%	**0.0±0.0%**
MinMIN-EFT	3.3±2.2%	72.5±45.4%	**0.0±0.0%**
NSGA-II	2.8±1.7%	71.2±44.2%	**0.0±0.0%**
MOCellSRF	2.8±1.7%	71.2±44.2%	**0.0±0.0%**

The results reported in Table 4.3 show that both MOEAs compute equally accurate schedules in terms of makespan, with an average value for GAP_M of 22.0% and a GAP_M value as low as 9.7% when solving instances of the class *Homogeneous-Parallel* using the MOCellSRF algorithm. Results show that *Series-Parallel* and *Heterogeneous-Parallel* are the hardest instance classes to tackle in terms of makespan, with MOEA computing solutions with an average value for GAP_M of 36.9%. On the other hand, *Single-Task*, *Homogeneous-Parallel*, and *Mix* instance classes are much easier to tackle, with MOEA computing solutions with an average value for GAP_M of 12.1%.

Regarding the energy consumption, the results show that both MOEA compute very accurate schedules—with higher accuracy than for the makespan objective—. The obtained values for GAP_E are as low as 7.8% when tackling *Homogeneous-Parallel* instances using MOCellSRF. Nevertheless, energy consumption is also the objective for which the heuristics are the less effective, with a value of GAP_E as high as 227.6%.

Finally, when analyzing the computed values for the service-level agreement objective, results show that all algorithms are able to compute the lower bound on every instance class with the exception of the *Single-Task* instance class. Nevertheless, when tackling this latter instance class all algorithms compute very accurate schedules with an average value for GAP_S of 1.1%.

Large problem instances

This subsection summarizes the results obtained for the large problem instances, using both the heuristics and the MOEAs.

Deterministic heuristics. Table 4.4 presents the results found by the heuristics described in Section 4.4.1. For makespan (f_M) and energy consumption (f_E) objectives, we computed the difference of the performance of every heuristic (in %) with respect to the best found result (among all studied heuristics) for every instance. In the table, we show the average of the computed difference for the 25 instances of every instance class. It is also shown in the table the average number of times in which the SLA was violated by the solutions proposed by the heuristics, averaged for the 25 instances of every class. The best results for every instance class are emphasized in **bold font**.

Table 4.4: Results obtained with the two-level deterministic heuristic schedulers

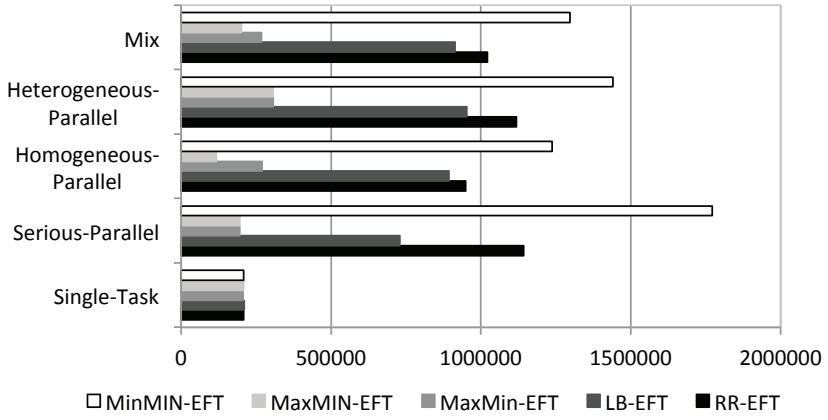
Single-Task			
Algorithm	f_M	f_E	f_S
RR-EFT	9.6 ± 12.5	2.3 ± 2.5	0.04 ± 0.20
LB-EFT	10.4 ± 10.5	9.3 ± 2.5	**0.00 ± 0.00**
MaxMin-EFT	6.8 ± 10.0	1.5 ± 2.1	0.21 ± 0.41
MaxMIN-EFT	**5.8 ± 8.5**	**1.3 ± 1.8**	0.13 ± 0.33
MinMIN-EFT	7.3 ± 10.2	3.4 ± 2.7	0.08 ± 0.28
Series-Parallel			
Algorithm	f_M	f_E	f_S
RR-EFT	5.7 ± 2.4	1.4 ± 0.5	7.64 ± 1.22
LB-EFT	3.9 ± 2.3	0.7 ± 0.5	11.80 ± 2.27
MaxMin-EFT	**0.3 ± 0.8**	**0.1 ± 0.2**	17.88 ± 1.13
MaxMIN-EFT	**0.3 ± 0.8**	0.3 ± 0.3	17.88 ± 1.13
MinMIN-EFT	8.6 ± 2.5	4.3 ± 0.8	**0.00 ± 0.00**
Homogeneous-Parallel			
Algorithm	f_M	f_E	f_S
RR-EFT	21.0 ± 17.5	1.7 ± 2.2	4.96 ± 1.81
LB-EFT	21.6 ± 18.9	9.0 ± 2.3	5.72 ± 2.26
MaxMin-EFT	12.7 ± 16.0	3.8 ± 2.1	15.80 ± 1.78
MaxMIN-EFT	**2.0 ± 4.9**	**0.6 ± 1.1**	18.20 ± 1.08
MinMIN-EFT	12.3 ± 11.6	8.8 ± 2.0	**0.00 ± 0.00**
Heterogeneous-Parallel			
Algorithm	f_M	f_E	f_S
RR-EFT	7.7 ± 3.3	1.7 ± 0.7	4.76 ± 1.33
LB-EFT	3.9 ± 3.6	0.9 ± 0.9	6.64 ± 2.29
MaxMin-EFT	**0.5 ± 1.3**	**0.2 ± 0.4**	15.84 ± 1.55
MaxMIN-EFT	**0.5 ± 1.3**	**0.2 ± 0.4**	15.84 ± 1.55
MinMIN-EFT	11.6 ± 3.8	5.3 ± 1.2	**0.00 ± 0.00**
Mix			
Algorithm	f_M	f_E	f_S
RR-EFT	15.8 ± 15.6	2.5 ± 2.0	4.44 ± 1.96
LB-EFT	20.0 ± 12.5	8.5 ± 3.2	5.80 ± 1.47
MaxMin-EFT	**7.3 ± 14.3**	5.2 ± 2.5	15.92 ± 2.14
MaxMIN-EFT	9.7 ± 12.9	**0.3 ± 0.7**	17.08 ± 1.73
MinMIN-EFT	20.7 ± 15.3	7.3 ± 2.0	**0.00 ± 0.00**

From the results in Table 4.4, we conclude that the best heuristic in terms of both makespan and energy consumption is MaxMIN-EFT, computing the best solutions for four out of the five studied instance classes. In terms of makespan, it is outperformed by MaxMin-EFT when tackling the Mix class of instances. Regarding the energy consumption, MaxMIN-EFT is only outperformed when tackling the Series-Parallel class of instances, again by MaxMin-EFT. Both heuristics offer the same average performance for Series-Parallel class of instances for these two objectives. Please, notice that due to the heterogeneity of our federated datacenter, the energy consumption is not directly related to the makespan of solutions. In this sense, comparing MaxMIN-EFT versus MaxMin-EFT, we can see how the former outperforms the latter in terms of energy consumption for Mix instances despite its worse makespan; on the contrary, MaxMIN-EFT finds worse solutions in terms of energy consumption with the same average makespan in Series-Parallel instances.

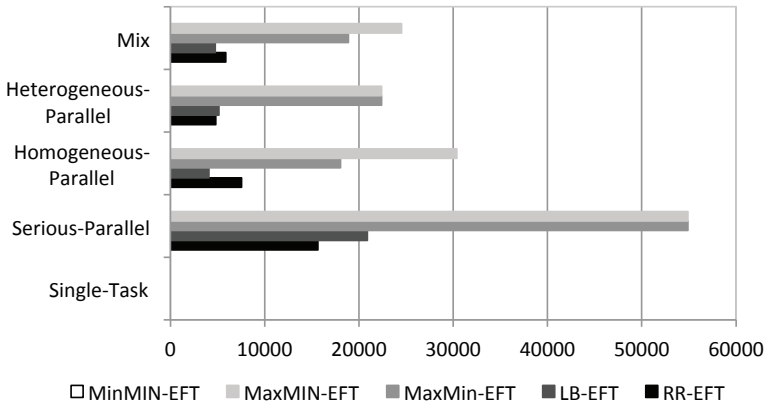
Regarding the number of times in which the SLA is violated, we can see that MinMIN-EFT heuristic is clearly the best algorithm, outperforming the others in all instance classes except for Single-Task, for which its performance is very close to the optimal value. The two best heuristics in terms of makespan and energy consumption, namely MaxMin-EFT and MaxMIN-EFT, are the worst ones according to the SLA agreement in every problem class.

We further analyzed the behavior of the heuristics in terms of the SLA. Figure 4.5(a) shows how far the solutions were from violating the SLA (*left over* metric), while Figure 4.5(b) reports by how much the SLAs were violated (*time exceeded* metric).

Figure 4.5(a) shows that MinMIN-EFT is the algorithm providing solutions with a larger margin of time units before the SLA is violated. This means that its solutions allow bigger delays than the solutions reported by the other heuristics without violating the SLA. Therefore, these solutions are more robust against some uncertainties that might occur during the execution of the workflows. Figure 4.5(b) shows that MinMIN-EFT is also the best algorithm according to the aggregated time exceeded for those schedules violating the SLA (its value is zero for all problem classes, except for Single-Task, for which it is almost zero). Therefore, MinMIN-EFT is clearly the best performing algorithm for the SLA objective.



(a) Left over time of schedules that do not violate the SLA



(b) Time exceeded over the SLA

Figure 4.5: Aggregated (a) left over time units of all solutions meeting the SLA and (b) time units over the SLA for those solutions that did not meet it.

Table 4.5: Results of the Friedman statistical test on the studied heuristics

Algorithm	*Ranking*
MaxMIN-EFT	2.20
MaxMin-EFT	2.60
RR-EFT	2.93
MinMIN-EFT	3.20
LB-EFT	3.53

In order to study if one single heuristic is better than the others according to its overall performance (for the three objectives), we applied the non-parametric Friedman statistical test to analyze the results distributions of the heuristics for every instance class, considering all the studied objectives. The Friedman test did not find statistical differences between the heuristics, as pointed out by the similar ranking values reported in Table 4.5. Hence, we applied the Holm-Bonferroni test to look for pairwise differences between the heuristics results on all problems and, again, no statistical differences were found in any comparison

The p-value of the comparison between MaxMIN-EFT and LB-EFT—i.e., the best and worst algorithms in the rank, respectively—obtained with the Holm-Bonferroni test was 0.43308. This results were somehow expected, since we are aggregating the results computed for all three objectives; those heuristics that perform better for makespan and energy objectives are the worst ones according to SLA, and the best ones for SLA offer the poorest results for the other objectives. Although there is no statistical confidence about the differences on the results computed by the deterministic heuristics over all problem instances, the results in Table 4.5 show that MaxMIN-EFT offered the best overall performance, having an aggregated average rank of 2.20 in all instance classes and problem objectives.

Table 4.6 reports the results of the two studied MOEAs for the different instance classes according to the three metrics described in section 4.5.2. We do not provide results for the instances on the SingleTask class, because the algorithms were not able to find more than two solutions in most cases, and the computation of the metrics is not possible in that case. A probable cause for this behavior is that the solution space is more sparse for this class of problem, making it difficult to the MOEAs to find a correct diversity in the non-dominated solutions.

Table 4.6: Comparison of the two multiobjective evolutionary algorithms by means of their average value and standard deviation for three different metrics (large problem instances)

workload type	HV	
	NSGA-II	MOCellSRF
Series-Parallel	$\mathbf{2.12\times10^{-1}\pm1.5\times10^{-1}}$	$1.88\times10^{-1}\pm1.3\times10^{-1}$
Homogeneous-Parallel	$\mathbf{9.03\times10^{-2}\pm1.1\times10^{-1}}$	$7.36\times10^{-2}\pm9.9\times10^{-2}$
Heterogeneous-Parallel	$2.12\times10^{-2}\pm6.7\times10^{-2}$	$1.44\times10^{-2}\pm5.2\times10^{-2}$
Mix	$\mathbf{2.03\times10^{-1}\pm1.4\times10^{-1}}$	$1.57\times10^{-1}\pm1.3\times10^{-1}$
workload type	Δ	
	NSGA-II	MOCellSRF
Series-Parallel	$8.34\times10^{-1}\pm7.6\times10^{-2}$	$\mathbf{7.14\times10^{-1}\pm5.8\times10^{-2}}$
Homogeneous-Parallel	$8.17\times10^{-1}\pm1.0\times10^{-1}$	$\mathbf{6.72\times10^{-1}\pm8.3\times10^{-2}}$
Heterogeneous-Parallel	$9.71\times10^{-1}\pm1.4\times10^{-1}$	$\mathbf{6.85\times10^{-1}\pm7.6\times10^{-2}}$
Mix	$7.51\times10^{-1}\pm8.2\times10^{-2}$	$\mathbf{6.47\times10^{-1}\pm6.0\times10^{-2}}$
workload type	$I^1_{\varepsilon+}$	
	NSGA-II	MOCellSRF
Series-Parallel	$4.60\times10^{2}\pm3.7\times10^{2}$	$4.48\times10^{2}\pm3.7\times10^{2}$
Homogeneous-Parallel	$1.34\times10^{3}\pm3.8\times10^{2}$	$1.36\times10^{3}\pm3.9\times10^{2}$
Heterogeneous-Parallel	$\mathbf{5.99\times10^{2}\pm2.5\times10^{2}}$	$6.37\times10^{2}\pm2.7\times10^{2}$
Mix	$1.03\times10^{3}\pm3.8\times10^{2}$	$1.07\times10^{3}\pm3.9\times10^{2}$

The results reported in Table 4.6 indicate that NSGA-II outperforms MOCellSRF in terms of HV, and there is statistical significance (according to the Wilcoxon statistical test) for all instance classes, but Heterogeneous-Parallel. However, MOCellSRF clearly outperforms NSGA-II in terms of diversity of solutions (Δ), with statistical significance in all cases. Finally, no statistical differences were found between the algorithms regarding the unary additive epsilon metric ($I^1_{\varepsilon+}$), with the only exception of Heterogeneous-Parallel instance class, for which NSGA-II outperformed its counterpart.

We conclude that both MOEAs compute accurate schedules, with MOCellSRF providing a more diverse set of solutions for the decision maker to choose from.

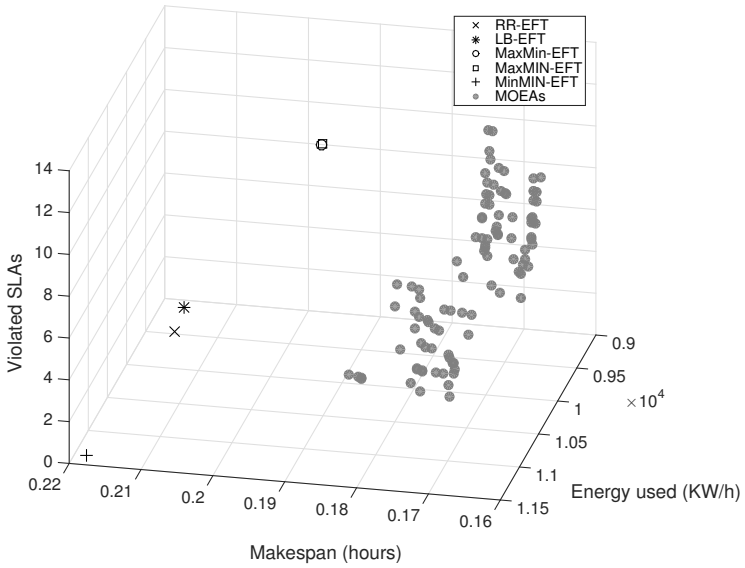
Figure 4.6 shows the solutions provided by the studied heuristics together with the obtained reference Pareto front for some sample instances of class Heterogeneous-Parallel and Mix, selected as they offer results that are representative of the ones computed for the other problem instances. The reference Pareto front for each instance is computed from all the solutions obtained by the two MOEAs in the 30 independent runs for that instance.

Among all non-dominated solutions, 100 representative ones are kept, chosen using the Strength Raw Fitness method. Figure 4.6 shows that the computed Pareto front offers a wide range of solutions to the problem, all of them providing makespan values that are significantly faster than the ones computed by the proposed heuristics.

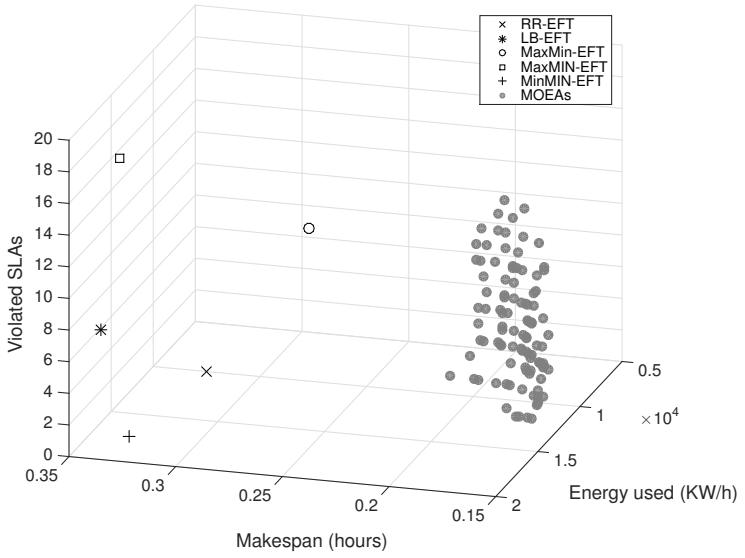
We computed the number of solutions from the reference Pareto fronts that dominate the solutions reported by all five heuristics (i.e., that are better or equal in all objectives and strictly better in at least one). We call them *dominating reference solutions*. As a result, we obtained that, in average for the 25 instances, 37.7%, 18.8%, 12.6%, 9.4%, and 7.5% solutions from the reference Pareto front dominate all the heuristics (i.e., are dominating reference solutions), for Single-Task, Series-Parallel, Homogeneous-Parallel, Heterogeneous-Parallel, and Mixed instance classes, respectively.

We studied the improvement of the dominating reference solutions with respect to the best value found using any heuristic for every objective, for every problem class. We find improvements of 39.36% and 49.02% (for makespan and energy consumption) for SingleTask, 0.45% and 3.13% for Series-Parallel, 16.05% and 36.80% for Homogeneous-Parallel, 24.25% and 56.91% for Heterogeneous-Parallel, and 21.62% and 62.27% for Mix. With respect to SLA, there is always some heuristic that finds a solution without any SLA violation in some of the 25 instances, therefore, it is obviously not possible for the MOEAs to improve that value.

The best improvements of the MOEAs were found for the SingleTask instances for makespan (by 39.36%) and for the Mix problems for energy consumption (by 62.27%). We also found that in all cases, the improvement was more important for energy consumption than for makespan in all cases. The lowest improvements were reported for the Series-Parallel instances, where there are fewer options for building different combinations of tasks for each workflow.



(a) Sample problem instance, Heterogeneous-Parallel workflow type



(b) Sample problem instance, Mix workflow type

Figure 4.6: Selected plots of the results provided by the heuristics and the MOEAs.

4.6 Conclusions

In this paper, we propose a novel multiobjective problem formulation to model the scheduling of a large number of workflows in a federated datacenter considering energy consumption and QoS. The main goal of the problem formulation is to simultaneously minimize three objectives: makespan, the energy consumption, and the number of jobs that violate the SLA agreement between the users and the service provider.

We present the design of a realistic solution based on hierarchical multiobjective schedulers to solve the problem. In the higher level of the hierarchy, an algorithm assigns workflows to the different datacenters, while in the lower level the local schedules of every assigned workflow are performed in each datacenter. We studied five heuristics to tackle the online version of the problem, and proposed two MOEAs to tackle the offline version of the problem. These MOEA were used to evaluate the performance of the five heuristics. Additionally, a constraint programming model was used to find lower bounds of medium-sized instances of a version of the problem with relaxed constraints.

Regarding the medium-sized instances, the numerical results obtained in the experimental evaluation—considering a benchmark set which includes different workflow types—show that Heterogeneous-Parallel is one of the hardest instance classes to tackle, but also one of the most promising in terms of room for accuracy improvement. For this class of instances, the MOEA compute solutions with an average of **40.0%** makespan gap and **39.3**% energy consumption gap, while the best heuristic for each objective compute an average gap of **57.9%** in makespan gap and **135.7**% in energy consumption. Regarding the large-sized instances, experimental evaluations confirms that Heterogeneous-Parallel is one of the hardest instance class, with the offline schedulers computing improvements of **24.2%** and **56.9%** in makespan and energy consumption over the best online methods.

When analizing the online schedulers, we can see that both MaxMin-based algorithms are able to compute the most accurate schedules for medium- and large-sized instances when considering all objectives. Nevertheless, both algorithms are also the less accurate schedulers in terms of SLA violations, with an average SLA violation gap of 13.5% for the large-sized instances. In this regard, the MinMIN-EFT heuristic is the most accurate scheduler, computing an average SLA violation gap of 0.0% also for the large-sized instances.

The main lines for future work include a further analysis of the proposed methods to improve both the offline and online schedulers. We propose to improve the efficiency of the method for computing the lower bound by reformulating and further relaxing the programming model. This will allow us to tackle larger medium-sized instances and compute the objective gaps of the MOEA solutions for larger instances. In order to improve the efficacy of the MOEA, we propose to include heuristic knowledge to the evolutionary algoritm, such as hybridizing a local search operator and using the proposed online schedulers to initialize the MOEA poulation. Finally, we propose to improve the efficacy of the online schedulers by hybridizing the most accurate heuristic schedulers in a fast local search algorithm.

Chapter 5

Energy aware multiobjective scheduling in a federation of heterogeneous datacenters

Santiago Iturriaga and Sergio Nesmachnow

Abstract: This work proposes three multiobjective evolutionary algorithms for energy aware scheduling in a federation of heterogeneous datacenters. The proposed algorithms schedule workflows of tasks aiming at optimizing infrastructure usage, quality of service and energy consumption. We extend our previous work by introducing a more realistic problem formulation and by performing a comprehensive statistical analysis for comparing the accuracy of three different evolutionary scheduling alternatives. We perform an extensive experimental evaluation with 100 problem instances, considering a diverse set of workflows and scenarios of diverse size. Results show that our newly proposed evolutionary approach is able to compute accurate schedules, outperforming our previously proposed approach as well as several traditional heuristic schedulers.

5.1 Introduction

This work presents the application of a two-level scheduling approach that combines a MultiObjective Evolutionary Algorithm (MOEA) and specific ad-hoc backfilling heuristics for energy-aware planning of workloads into a federation of heterogeneous distributed datacenters, taking into account task's dependen-

cies and quality of service (QoS) provided by the datacenter. The reported research extends the approach presented in Chapter 4 by considering a fully heterogeneous approach for both workloads and computing elements, a realistic assumption for nowadays high performance computing infrastructures (Ren et al., 2015). Furthermore, we extend our experimental analysis by designing and studying three different high-level MOEA schedulers and several high-level heuristic schedulers. The experimental analysis is performed over a set of 100 realistic problem instances considering both small- and large-sized scenarios. These problem instances are comprised of five different types of computing workflows, sampling a wide range of realistic high-performance applications. Each problem instance, disregarding its type, is comprised of 1000 computing workflows, each of which is comprised of a number of tasks ranging from 1 up to 132.

The main results indicate that the proposed MOEA outperforms the most accurate greedy heuristics with a makespan improvement of 32%, energy consumption improvement of 6%, and QoS improvement of 29%.

The article is organized as follows. Section 5.2 presents the problem formulation and reviews related works. The scheduling approach and the proposed MOEA to solve the problem are described in Section 5.3. The experimental evaluation is reported in Section 5.4, including a comparative analysis of the proposed methods and a comparison against traditional schedulers. Finally, Section 5.5 presents the conclusions and the main lines for future work.

5.2 Modeling energy-aware scheduling in heterogeneous datacenters

The energy-aware scheduling problem proposes the allocation of resources to parallel tasks that must be executed on a federation of heterogeneous datacenters. The problem model for heterogeneous datacenters is presented next, considering the following elements:

- A distributed federation comprised by k heterogeneous datacenters $DC = \{dc_1, \ldots, dc_k\}$. Each datacenter DC_r comprised of a set of heterogeneous multi-core servers $S_r = \{s_1, \ldots, s_s\}$ organized in racks. All servers inside a rack are identical but servers may be different among racks. Each server s_j is characterized by its number of cores c_j,

its performance in FLOPS ops_j, and the power consumption at idle e_j^{idle} and peak e_j^{max} utilization. Servers in each rack are networked together through a top-of-the-rack (TOR) switch. This TOR switch networks together all servers in its rack allowing a communication speed of rs_j, while an aggregation switch networks all the TOR switches with a communication speed of as_j. Communication between processes in the same server are considered to be instantaneous similar to the approaches proposed by Chen et al. (2016a); Prajapati and Shah (2014); Sharifi et al. (2013); Wang et al. (2016); Wu et al. (2016); and Zhu et al. (2016).

- A set of n independent heterogeneous workflows $Q = \{q_1, \ldots, q_n\}$. Each workflow q has an associated soft deadline d_q before it should be accomplished. Each workflow q is a parallel application decomposed into a set of tasks $WT_q = \{wt_1, \ldots wt_m\}$ with dependencies. Each task has different computing requirements. After a task finishes its execution it produces an output dataset that is input to all of its successor tasks. A task wt_α is eligible for execution in server s_β only after all its predecessors have finished executing and the dataset generated by them has been transferred to server s_β.
- A number p of workflow owners $O = \{o_1, \ldots, o_p\}$ and a SLA that determines the percentage of applications that should be finished before their deadlines.
- Each task wt_α is characterized by the tuple $(o_\alpha, nc_\alpha, d_\alpha)$ defining its length (number of operations), the number of resources (cores) required for the parallel execution, and the networking time required for transferring its output dataset.

We propose to optimize two system-related objectives that take into account the point of view of the computing system, and a QoS-related objective that takes into account the point of view of the users. These objectives are the same we introduced in our previous formulation (see Section 4.2) and are described by the following functions:

- *makespan* evaluates the total time to execute a set of workflows, according to the expression in Eq. (5.1), where $\vec{x}$ represents an allocation, k is the number of available datacenters, and CT_r is the completion time of datacenter r (DC_r).

$$f_M(\vec{x}) = \max_{0 \leq r \leq k} CT_r \tag{5.1}$$

- *energy consumption* for a set of workflows executed in a certain datacenter, defined by Eq. (5.2), using the energy model for multi-core architectures proposed by Nesmachnow et al. (2013), where f_1 is the higher-level scheduling function, and f_2 is the lower-level scheduling function. The total energy consumption takes into account both the energy required to execute the tasks assigned to each computing resource within a datacenter, and the energy that each resource consumes in idle state.

$$f_E(\vec{x}) = \sum_{r \in DC} \sum_{\substack{q \in Q: \\ f_1(q)=r}} \sum_{\substack{wt_\alpha \in WT_q: \\ f_2(wt_\alpha)=s_j}} \frac{o_\alpha}{ops(s_j)} \times e_{s_j}^{max} + \sum_{s_j \in S_r} e_{s_j}^{idle} \tag{5.2}$$

- *SLA violations* is defined as the number of workflows that do not finish before their deadline, over the allowed limit specified by the SLA, according to Eq. (5.3), where $V(q)$ is 1 when the deadline of workflow q is violated and 0 otherwise, and $W(u_i)$ is the number of workflows submitted by user u_i.

$$f_S(\vec{x}) = \sum_{u_i \in U} \max \left(0, \sum_{q \in wo(u_i)} V(q) - (1 - SLA_{u_i}) \times W(u_i) \right) \tag{5.3}$$

5.3 The proposed two-level multiobjective evolutionary schedulers

In this section we present three different MOEA for addressing the proposed scheduling problem. All three of them apply the low-level EFTH heuristic scheduler presented in Chapter 4.

The first high-level scheduler is based on the *Non-dominated Sorting Genetic Algorithm, version II* (NSGA-II) (Deb, 2001). NSGA-II is a popular state-of-the-art MOEA that showed to be accurate for solving the problem formulation presented in Chapter 4. A detailed description of the workings of NSGA-II is presented in Section 4.3.2. The second scheduler is based on the *Indicator-Based Evolutionary Algorithm* (IBEA) (Zitzler and Künzli, 2004) using an hypervolume-based indicator. Hypervolume is a well-known metric that measures both diversity sampling and convergence to the Pareto front, providing an accurate Pareto-compliant measure. IBEA applies the hypervolume-based indicator for computing the fitness of each candidate solution and simply ranking solutions according to their fitness. Since fitness is computed using a

Pareto-compliant indicator, IBEA does not require any mechanism for preserving diversity such as fitness sharing or crowding. Finally, the third scheduler is based on the $\mathscr{S}$ *Metric Selection Evolutionary Multiobjective Optimisation Algorithms* (SMS-EMOA) (Beume et al., 2007). The $\mathscr{S}$ metric is also known as the hypervolume metric, the same metric used for IBEA. However, SMS-EMOA applies the hypervolume metric differently than IBEA. SMS-EMOA uses a non-dominated ranking criterion for ranking solutions into fronts (just like NSGA-II), and applies hypervolume as a removal criterion for preserving diversity. That is, each generation the solution that contributes the least to the hypervolume of the worst front is removed. Both NSGA-II and IBEA use a generational evolutionary approach where for each generation a whole new population of offspring solutions is created. The solutions from this new population compete with the solutions of the old population for surviving to the next generation. On the contrary, SMS-EMOA uses a steady-state approach where just one solution is created each generation, and this newly created solution competes for survival with the worst solution of the current population. Next, we present the main features of the NSGA-II, IBEA and SMS-EMOA high-level schedulers.

Solution encoding. We use the solution encoding presented in Chapter 4. This encoding proved to be efficient and is able encode the problem formulation proposed in this work without any modification. In this encoding, solutions are represented as a vector of integers ranging from 0 to $n + k - 1$, with n the number of workflows and k the number of datacenters. Integers in [0, $n - 1$] represent workflows while integers in [n,$n + k - 1$] act as a separators of workflows assigned to each datacenter.

Fitness values. We define three objective functions, exactly as defined in Section 5.2. NSGA-II and SMS-EMOA evaluate the quality of a solution according to a ranking based on non-domination sorting. NSGA-II applies a crowding distance metric for preserving diversity, while SMS-EMOA uses the hypervolume metric. IBEA considers a different approach and defines a single fitness value based on the hypervolume metric. Eq. 5.4 presents the fitness function f proposed by IBEA, with $I_{HD}(x_2, x_1)$ representing the difference between the hypervolume computed for solutions x_1 and x_2 in population P. IBEA uses no diversity preserving mechanism.

$$f(x_1) = \sum_{x^2 \in P \setminus \{x_1\}} -e^{-I_{HD}(x_2,x_1)/0.05} \tag{5.4}$$

Population initialization. The initial population is created by applying a set of high-level heuristic schedulers. These heuristics are applied in random order until all initial solutions are created. In Section 5.4.2 we describe in detail the workings of the considered high-level heuristic schedulers. This initialization method is applied the same for all MOEA.

Selection. NSGA-II and IBEA use binary tournament as the selection operator, taking into account Pareto dominance and crowding distance. This operator works by randomly selecting two solutions from the population. If one of the selected solution dominates the other, then that solution is selected and the other discarded. Otherwise, one of the two solutions is randomly selected. SMS-EMOA uses a simple random selection strategy where a solution is randomly selected from the population.

Variation operators. All MOEA apply the Partially Matched Crossover (PMX) and Exchange Mutation (EM) for combining and mutating solutions. PMX works by selecting two random positions in the selected solutions and swapping all values between them. The remaining values are rearranged using position wise exchanges, maintaining the ordering information. EM operator is much simpler than PMX and works by randomly selecting two values in a solution vector and swapping them.

Repair operator. This heuristic operator transforms non-feasible solutions into feasible ones. It is used by all MOEA to correct solutions after applying variation operators. It works by checking if each workflow is assigned to a datacenter that is capable of executing it. If not, then the solution is repaired by reassigning unfeasible workflows to random datacenters that can execute them.

Parameter configuration. All the MOEA were configured with a population size of 100, stopping criterion of 25000 evaluations, crossover probability of 0.9, and mutation probability of $1/n$ (with n the number of workflows).

5.4 Experimental evaluation

This section presents the experimental evaluation of the proposed scheduling methods.

5.4.1 Problem instances

Each problem instance is defined by a *workload* of workflows and a *scenario* of available computing resources. This work applies the five workflow models presented in Chapter 4: Series-Parallel, Heterogeneous-Parallel, Homogeneous-Parallel, Single-Task, and Mix. The SchMng method, proposed by Taheri et al. (2013), is used for creating all workflow instances. The deadline d_q of each workflow q is randomly generated by extending the completion time of the critical path of q by a ratio of $[0.05, 0.30]$. This is a realistic ratio according to Garg and Singh (2016) and Tang et al. (2016). Following a similar approach, networking communication time d_α for each task wt_α is randomly generated as a ratio of $[0.05, 0.50]$ of the execution time of wt_α, similar to the ratios proposed by Chen et al. (2016a) and Tang et al. (2016).

We consider each data center to be organized in racks of servers, each rack containing between 18–42 servers, with processors in the same rack being homogeneous. However, different racks may contain different types of processors. Processors in each rack are randomly chosen from a set of modern Intel processors ranging from 1 to 6 cores each. The candidate processors are presented in Section 4.5.1 and detailed in Table 4.2. We define three realistic service level agreements, $SLA = \{90\%, 94\%, 98\%\}$. Each SLA represent the minimum ratio of workflows per user that must meet their deadlines. These are realistic values provided by current datacenter and cloud computing facilities as presented in Chapter 4.

A total of 50 workflow batches were created for the experimental evaluation, 10 for each of the 5 workflow types. Each batch is comprised of a total of 1000 workflows, with the number of tasks in each workflow ranging from 3 to 132, except for the Single-Task workflows that are comprised of just one task. In total, 50000 workflows are studied in the experimental analysis.

For the datacenter scenario we consider small- and large-sized scenarios. Both involving a federation of five datacenters with up to three racks each. The small-sized scenario is comprised by an average of 150 processors per datacenter, and the large-sized instance by an average of 325 processors.

Racks are communicated by 1GB or 10GB Ethernet networks.

Overall, 100 problem instances were evaluated, considering all workflows and scenarios. The benchmark set of workflows, scenarios, and SLA levels is publicly available, it can be accessed/downloaded by contacting the authors.

5.4.2 High-level scheduling heuristics

We consider seven different heuristic scheduling algorithms. These algorithms work by iteratively applying greedy decisions until a full schedule is constructed. Each iteration a workflow is selected and scheduled following some heuristic knowledge that varies for each heuristic. This process is repeated until all workflows are scheduled. A short description of the scheduling algorithms is presented next.

1. *Round Robin* (RR): Given a list of workflows and datacenters, this algorithm assigns the first workflow to the first datacenter, the second workflow to the second datacenter, and so on. Once the last datacenter is reached, the assignment continues with the first datacenter, following a circular strategy. If a datacenter is unable to satisfy a workflow requirements, then it is skipped and the workflow is assigned to the next datacenter suitable for it.
2. *Load Balance* (LB): This algorithm aims for a balanced workflow assignment. To accomplish this, workflows are first sorted according to the maximum number of cores required by any of their tasks. This way, workflows requiring more cores are prioritized for scheduling. Workflows are scheduled one at a time in the sorting order to the datacenter with the lowest number of assigned workflows.
3. *MaxMin*: This algorithm assigns each workflow to the datacenter that can execute it faster. In each iteration, the workflow with the maximum estimated completion time is selected and assigned to a datacenter that can finish it earlier, considering its core requirements. In this case, the completion time of each workflow is estimated with the sum of the execution time of the tasks in its critical path. This procedure continues until all workflows are assigned. As a result, longer workflows are scheduled before shorter workflows, following the heuristic knowledge that scheduling longer workflows first produce a more balanced schedule.

4. *MaxMIN*: Just like MaxMin, this algorithm schedules first the workflows with longest completion estimation time, considering the completion estimation time of all the other workflows scheduled in the datacenters. However, workflows are scheduled to the datacenter that minimizes their overall energy consumption estimation. That is, MaxMIN schedules the longest workflows first to the datacenter consuming the minimum amount of energy. Again, energy consumption estimation and execution time estimation in this heuristic is based on the total execution time of the tasks in its critical path.
5. *MinMIN*: Similar to MaxMIN, but in this case in each iteration of the algorithm, the workflow that requires the overall minimum completion time is selected first and assigned to a datacenter with the lowest energy consumption.
6. *Core-Aware MaxMin* (CA-MaxMin): The CA-MaxMin works exactly as MaxMin, but estimates the overall completion time with the execution time of the tasks in its critical path multiplied by the number of cores required by all of its tasks. This new estimator prioritizes the scheduling of workflows with heavy processing requirements.
7. *Longest First* (LF): Finally, the LF is a simple algorithm based on the LB heuristic. Both algorithm are identical, except initial workflow sorting is performed differently. The LF heuristic sort workflows are sorted considering the product of their critical path execution time, the sum of the execution time of all of the tasks in the workflow, and the sum of the total number of cores required by all of its tasks. This estimator schedules first the most computing demanding workflows, balancing these workflows adequately among the data centers.

The RR, LB, MaxMin, MaxMIN, and MinMIN algorithms were all introduced in Chapter 4, while CA-MaxMin and LF were newly designed for this work. Furthermore, because the EFTH low-level heuristic does not deal with workflow ordering, we propose to combine the proposed high-level heuristics with a workflow sorting algorithm. That is, after the scheduling algorithm assigns each workflow to a datacenter, a sorting algorithm optimizes the ordering of workflows in each datacenter. The rationale for this is that the best ordering for the assignment of workflows to datacenters may not be the same as the best ordering for the execution of the workflows in each datacenter.

The considered sorting criteria are the following:

1. *Unsorted* (U): Applies no sorting algorithm. Workflows remain in the order the workflow assignment process produced.
2. *Average Cores per level* (AC): Sorts workflows according to the average number of total cores per level of the workflow graph.
3. *Maximum Cores* (MC): Sorts workflows according to the number of cores required by the task that requires the most cores. Sorting is untied by considering the length of the critical path of the workflow.
4. *Computing Load* (L): Sorting is performed based on the total execution time of the tasks in the critical path of the workflow multiplied by the average number of cores per level of the workflow graph.

Overall, we considered a total of 56 high-level scheduling heuristics, considering the combination of each scheduling algorithm with each sorting criterion in ascending (ASC) and descending (DSC) direction. For simplicity, from now on heuristics will be referred using the nomenclature: *scheduling algorithm + sorting criterion + sorting direction*. For example, the heuristic comprised by the RR algorithm, with AC sorting in ascending direction is referred as: RR+AC+ASC.

5.4.3 Experimental setup

This section details the experimental methodology applied in this work. Because of their stochastic nature, a total of 30 independent executions of each MOEA where performed for each of the 100 problem instances. Hence, a total of 3000 executions were performed for each MOEA. The results of these independent executions are compared with the following multiobjective metrics.

- Hypervolume (HV) (Zitzler and Thiele, 1999): this metric measures the the m-dimensional volume in the objective space of the solutions comprising the Pareto front Q relative to a reference point w. This volume is computed as following. For each solution $i \in Q$, a hypercube v_i is constructed with the reference point w and the solution i as the diagonal corners of the hypercube. The hypervolume is the volume of the union of all hypercubes as shown in Eq. (5.5). This metric takes its

maximum value when all the solutions of the optimal Pareto front are in Q, hence algorithms with highest HV value are preferred.

$$HV(Q) = \text{volume}\left(\bigcup_{i=1}^{|Q|} v_i\right) \tag{5.5}$$

- Unary Additive Epsilon (*epsilon*) (Knowles et al., 2006): this metric measures the distance from the Pareto front Q to the optimal Pareto front. For each solution $i \in Q$ it measures the smallest distance needed to translate it so that it dominates the optimal Pareto front Q^*. That is, given $\vec{i} = (i_1, ..., i_m)$ and $\vec{i^*} = (i_1^*, ..., i_m^*)$ where m is the number of objectives, $\vec{i} \in Q$, and $\vec{i^*} \in Q^*$, we define epsilon as shown in Eq. 5.6.

$$epsilon(Q) = \inf_{\varepsilon \in \mathbb{R}} \{\forall \vec{i^*} \in Q^*, \exists \vec{i} \in Q : \vec{i} + \varepsilon \succ \vec{i^*}\} \tag{5.6}$$

- Inverted Generational Distance (IGD) (Coello Coello and Reyes Sierra, 2004): this metric is the normalized sum of the distances between a set of uniformly distributed solutions in the optimal Pareto front $i^* \in Q^*$ and their closest solution in the computed Pareto front $i \in Q$. It is defined in Eq. 5.7, where $d(i^*, Q)$ is the Euclidean distance from solution i^* to the closest solution in the computed Pareto front.

$$IGD(Q) = \frac{1}{|Q^*|} \sum_{i^* \in Q^*} d(i^*, Q) \tag{5.7}$$

- Spread (Deb et al., 2002): given a Pareto front Q, this metric measures the spreading of its solutions $i \in Q$ in the objective space. It is defined by Eq. (5.8), where d_i is the Euclidean distance between consecutive solutions, $\bar{d}$ is the mean of all these distances, and d_f and d_l are the Euclidean distances to the most extreme solutions of the optimal Pareto front. This metric takes a zero value for an ideal distribution, pointing out a perfect spread of the solutions in Q.

$$spread(Q) = \frac{d_f + d_l + \sum_{i \in Q} \left| d_i - \bar{d} \right|}{d_f + d_l + |Q|\bar{d}} \tag{5.8}$$

All the presented metrics require some characteristic of the optimal Pareto front of the problem. However, this Pareto front is unknown and almost im-

possible to compute for realistic instances. Hence, in this work we approximate the optimal Pareto front for each problem instance with the best known Pareto front comprised by the union of the non-dominated solutions of all executions of all MOEA.

Regarding the statistical analysis. The Kolmogorov-Smirnov test (Sprent and Smeeton, 2000) is applied for testing normality. If normality is rejected, then the nonparametric Kruskal-Wallis one-way analysis of variance (Kruskal and Wallis, 1952) is applied for finding significant differences between algorithms. When significant differences are found, the nonparametric Dunn's post hoc test (Dunn, 1961) is used for pinpointing pairwise differences between algorithms. Since normality is rejected for all samples, parametric tests are not considered. Overall, we say there is statistical significance if $p-value \leq 0.05$.

5.4.4 Development and execution platform

The proposed high-level heuristics were implemented Java. The high-level MOEA was also implemented in Java using the jMetal framework (Durillo and Nebro, 2011). The low-level heuristic was implemented in C and compiled using the GNU C compiler. All the experiments were performed in ClusterFING, the HPC facility of Universidad de la República, Uruguay (platform website: `https://www.fing.edu.uy/cluster`).

5.4.5 Numerical results

This section reports the numerical results computed by the proposed methods. First, all high-level heuristic schedulers are studied and the most accurate heuristic for each objective is selected for comparing with the high-level MOEA schedulers. After that, the high-level MOEA schedulers are studied and the most accurate MOEA is compared with the most accurate heuristics.

High-level heuristic schedulers.

This section presents a comparison between the proposed high-level scheduling heuristics and studies the best heuristics for optimizing each objective. Since these schedulers are single-objective algorithms, the comparison is performed separately for each problem objective using the gap metric.

The gap metric of a given objective for some scheduler is defined as the unity-based normalization of the objective value. Eq. 5.9 presents the gap metric for scheduler s, with v_s the value for the objective computed by s, and v_w the worst value and v_b the best value computed by any scheduler. Because all considered objectives are minimization objectives, the smaller the gap value the better the result. That is, when $gap = 0$ then s is the scheduler computing the best schedule for that objective, and when $gap = 1$ then s is the scheduler computing the worst schedule. We define the gap_M, gap_E, and gap_S for measuring the gap of the makespan, energy consumption, and SLA violations objectives respectively.

$$gap = \frac{v_s - v_b}{v_w - v_b} \tag{5.9}$$

We perform a statistical analysis over the results of the proposed scheduling heuristics for determining the most accurate heuristics for each objective. First, normality of the computed values is rejected with $p{-}value \leq 0.0077$ by applying a Kormogorov-Smirnov test for normality. After rejecting normality, the Kruskal-Wallis test is applied to test the equality of the medians of the results. The Kruskal-Wallis test shows significant differences between the medians with a $p{-}value \leq 0.0001$. Hence, there are significant differences in the accuracy of at least one of the heuristics. The Dunn's post hoc test shows no heuristic is significantly more accurate than all the rest for any objective. However, a ranking of the pairwise differences (with $p{-}value \leq 0.05$) shows the overall best performing scheduling algorithms are CA-MaxMin and LF.

Table 5.1 presents the average and standard deviation gap computed by the best performing scheduling heuristics for each problem objective, with the most accurate results presented in gray. The Dunn's test shows CA-MaxMin+U, CA-MaxMin+AC+ASC, CA-MaxMin+MC+ASC, and CA-MaxMin+L+ASC are all significantly more accurate than around 84% of the remaining heuristics when optimizing makespan. LF+AC+DSC and LF+MC+DSC are significantly more accurate than around 87% of the remaining heuristics when optimizing the energy consumption objective. And finally, CA-MaxMin+L+DSC is significantly more accurate than around 70% of the remaining heuristics when optimizing the SLA violations.

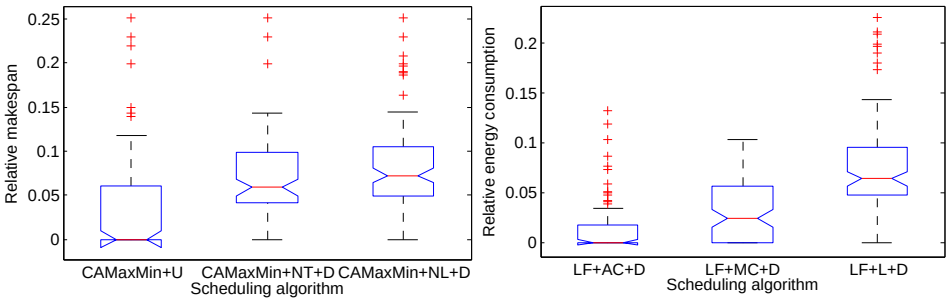
Table 5.1: Average and standard deviation gap values of the most accurate high-level scheduling heuristics for all the problem instances.

scheduling algorithm	*sorting criterion*	*sorting direction*	gap_M	gap_E	gap_S
MaxMin	U	none	0.10±0.09	0.67±0.07	0.69±0.13
	AC	ASC	0.10±0.09	0.67±0.07	0.69±0.13
		DSC	0.21±0.07	0.71±0.07	0.25±0.07
	MC	ASC	0.10±0.09	0.67±0.07	0.69±0.13
		DSC	0.27±0.07	0.74±0.08	0.31±0.08
	L	ASC	0.10±0.09	0.67±0.07	0.69±0.13
		DSC	0.23±0.08	0.78±0.06	0.06±0.04
MinMIN	U	none	0.35±0.14	0.70±0.10	0.19±0.08
	AC	ASC	0.35±0.14	0.70±0.10	0.19±0.08
		DSC	0.39±0.12	0.66±0.11	0.23±0.06
	MC	ASC	0.35±0.14	0.70±0.10	0.19±0.08
		DSC	0.45±0.11	0.68±0.10	0.31±0.06
	L	ASC	0.35±0.14	0.70±0.10	0.19±0.08
		DSC	0.39±0.12	0.71±0.10	0.11±0.05
CA-MaxMin	U	none	0.03±0.04	0.63±0.06	0.76±0.16
	AC	ASC	0.03±0.04	0.63±0.06	0.76±0.16
		DSC	0.09±0.04	0.65±0.06	0.23±0.07
	MC	ASC	0.03±0.04	0.63±0.06	0.76±0.16
		DSC	0.14±0.05	0.68±0.07	0.31±0.08
	L	ASC	0.03±0.04	0.63±0.06	0.76±0.16
		DSC	0.12±0.04	0.71±0.06	0.05±0.04
LF	U	none	0.90±0.08	0.16±0.08	0.90±0.06
	AC	ASC	0.90±0.08	0.16±0.08	0.90±0.06
		DSC	0.89±0.07	0.01±0.02	0.71±0.10
	MC	ASC	0.90±0.08	0.16±0.08	0.90±0.06
		DSC	0.97±0.04	0.03±0.02	0.78±0.11
	L	ASC	0.90±0.08	0.16±0.08	0.90±0.06
		DSC	0.92±0.05	0.08±0.04	0.65±0.07

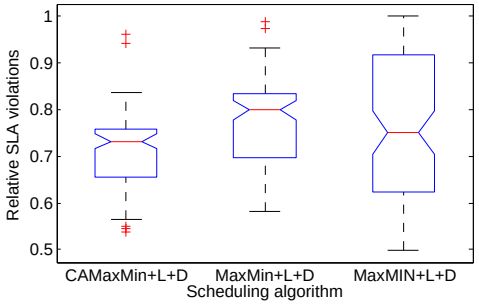
Best results for each objective are marked in gray ($p-value \leq 0.05$).

Several heuristics showed to be equally accurate for optimizing the makespan objective. Hence, following the Occam razor principle we select CA-MaxMin+U for the comparison with the MOEAs. Again, LF+AC+DSC and LF+MC+DSC showed no significant difference between them for optimizing energy consumption. However, LF+AC+DSC shows a better average gap metric than LF+MC+DSC. Hence, LF+AC+DSC is selected for comparison with the MOEAs. Finally, CA-MaxMin+L+DSC is selected for comparing of the SLA violations objective with the MOEAs. Figure 5.1 shows the relative

values of each objective computed by the most accurate heuristic schedulers for all instances. For simplicity, from now on the most accurate heuristics for each objective will be simply referred as *Makespan heuristic*, *Energy heuristic* and *SLA heuristic* respectively.



(a) Relative makespan **(b)** Relative energy consumption



(c) Relative SLA violations

Figure 5.1: Relative makespan, energy consumption and SLA violations values computed by the most accurate heuristic schedulers for all instances.

High-level MOEA schedulers.

This section summarizes the study of the proposed MOEA schedulers. The study follows the methodology presented in the previous section. Normality is tested individually for each algorithm, each problem instance and each metric, using the Kolmogorov-Smirnov test. Results from this test reject normality with $p{-}value \leq 0.0001$. Hence, the Kruskal-Wallis test is applied for comparing the proposed algorithms for each problem instance and each metric. Again, the null hypothesis is rejected for the Kruskal-Wallis test with $p{-}value < 0.0004$, implying samples do not originate from the same distribution and significant differences exist between algorithms. Finally, the Dunn's test is applied for the pairwise comparison of the MOEA considering all multi-objective metrics. On the one hand, results show SMS-EMOA is able to sig-

nificantly outperform IBEA and NSGA-II on most problem instances when considering the hypervolume, epsilon and IGD metrics. On the other hand, NSGA-II is significantly more accurate than SMS-EMOA and IBEA for the spread metric on all problem instances. Table 5.2 shows the relative number of problem instances each algorithm is the most accurate for each metric with the best results presented in gray.

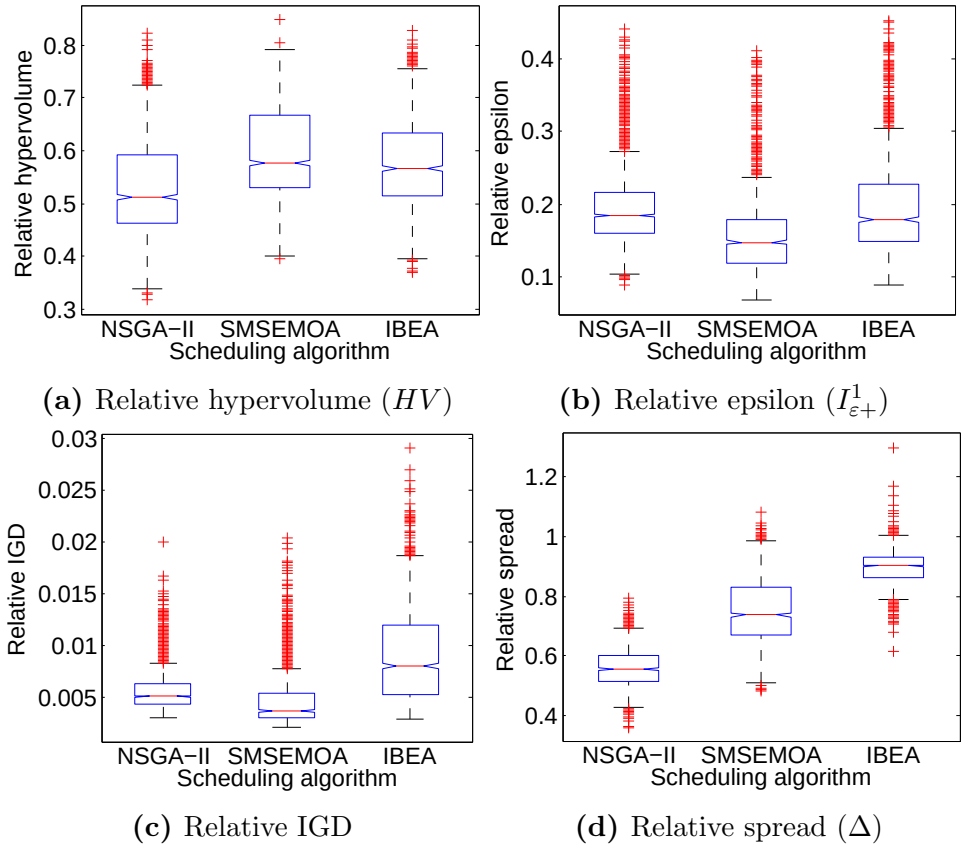
Table 5.2: Number of problem instances (out of 20) for which each MOEA is the most accurate according to each metric for small- and large-sized scenarios.

instance type	*algorithm*		
	NSGA-II	*IBEA*	*SMS-EMOA*
HV			
Heterogeneous-Parallel	0	16	20
Homogeneous-Parallel	0	15	20
Series-Parallel	0	12	19
Single-Task	4	2	9
Mix	0	16	20
epsilon			
Heterogeneous-Parallel	0	2	14
Homogeneous-Parallel	0	2	16
Series-Parallel	0	0	13
Single-Task	4	3	1
Mix	1	7	18
IGD			
Heterogeneous-Parallel	0	0	20
Homogeneous-Parallel	5	0	20
Series-Parallel	0	0	19
Single-Task	19	0	20
Mix	8	0	20
spread			
Heterogeneous-Parallel	20	0	2
Homogeneous-Parallel	20	0	0
Series-Parallel	20	0	8
Single-Task	20	0	13
Mix	20	0	0

Pairwise comparison with $p-value \leq 0.05$. Best results shown in gray.

Figure 5.2 show the relative hypervolume, epsilon, IGD and spread metrics for each MOEA considering all problem instances. Overall, these results indicate SMS-EMOA is the most accurate MOEA, computing the most accurate

solutions and sampling the most diverse Pareto front. Even though NSGA-II is able significantly outperform SMS-EMOA when considering the spread metric, it is clearly outperformed by SMS-EMOA according to the hypervolume, epsilon and IGD metrics.



(a) Relative hypervolume (HV)

(b) Relative epsilon ($I^1_{\varepsilon+}$)

(c) Relative IGD

(d) Relative spread (Δ)

Figure 5.2: Relative multiobjective metrics computed by the proposed MOEA for all instances.

Comparison of high-level schedulers.

This section studies and compares the solutions computed by the best heuristics and by SMS-EMOA. It compares the best heuristics and SMS-EMOA on an objective basis and it presents some visual representations of sample Pareto fronts and heuristics results.

Table 5.3 shows the average improvement and standard deviation for the best solution computed by SMS-EMOA for each objective when compared with the solution obtained by the best heuristic for that objective. Overall results show the SMS-EMOA outperforms the best heuristics, improving–in average–

the best heuristics by up to 32% in makespan, 6% in energy consumption and 29% in SLA violations. Nevertheless, there is significant variations in the accuracy of the SMS-EMOA results depending on the instance type and scenario size.

Table 5.3: Average and standard deviation of improvement of the SMS-EMOA scheduler for each objective when compared to the best heuristic scheduler

instance type	*improvement over the best heuristic*		
	f_M	f_E	f_S
small-sized scenarios			
Heterogeneous-Parallel	13.0±4.5%	9.5±0.6%	-1.2±1.4%
Homogeneous-Parallel	15.0±3.7%	12.0±0.6%	-4.1±2.9%
Serial-Parallel	12.0±5.3%	9.2±0.3%	0.4±1.1%
Single-Task	47.0±5.2%	7.3±0.6%	87.0±7.7%
Mix	12.0±2.3%	12.0±2.9%	-2.1±1.7%
large-sized scenarios			
Heterogeneous-Parallel	45.0±3.0%	2.6±0.9%	4.9±2.3%
Homogeneous-Parallel	37.0±9.1%	1.3±0.4%	63.0±9.4%
Serial-Parallel	47.0±3.3%	2.1±0.7%	4.0±2.3%
Single-Task	51.0±8.9%	1.4±0.8%	99.0±1.1%
Mix	37.0±5.7%	3.2±1.2%	41.0±25.0%

For the small-sized scenarios, SMS-EMOA produces the best makespan and SLA violations improvements when dealing with Single-Task instances, and produces the best energy consumption improvements when dealing with Homogeneous-Parallel instances. Furthermore, SMS-EMOA is able to consistently improve makespan and energy consumption objectives for all instances. However, although SMS-EMOA improves SLA violations by up to 87% when dealing with Single-Task instances, it is unable to improve or even worsens SLA violations for the remaining type of instances.

SMS-EMOA behaves differently when dealing with large-sized scenarios. For these scenarios, it is able to improve makespan consistently, with an average improvement of 37%–51% for all instances. Furthermore, SLA violations are also consistently improved on all instances, no longer worsening the heuristics results, and largely improving SLA violations for the Homogeneous-Parallel instances when compared to small-sized scenarios. However, improvement on energy consumption drops from 10% in average for small-sized scenarios to 2% in average for large-sized scenarios.

Overall, results show SMS-EMOA consistently improves heuristics results for the Single-Task instances in both scenarios and for all objectives, while the remaining type of instances present diverse results depending on the objective. On the one hand, SMS-EMOA consistently improves makespan in all scenarios. On the other hand, improvements of energy consumption and SLA violations are not consistent. Energy consumption is most improved for small-sized scenarios, with little improvement for large-sized scenarios. While SLA violations are most improved for large-sized scenarios, with negligible improvements for small-sized scenarios except when dealing with Single-Task instances. Figure 5.3 present samples of Pareto fronts computed by SMS-EMOA and schedules computed by each of the best heuristics for the large-sized scenarios.

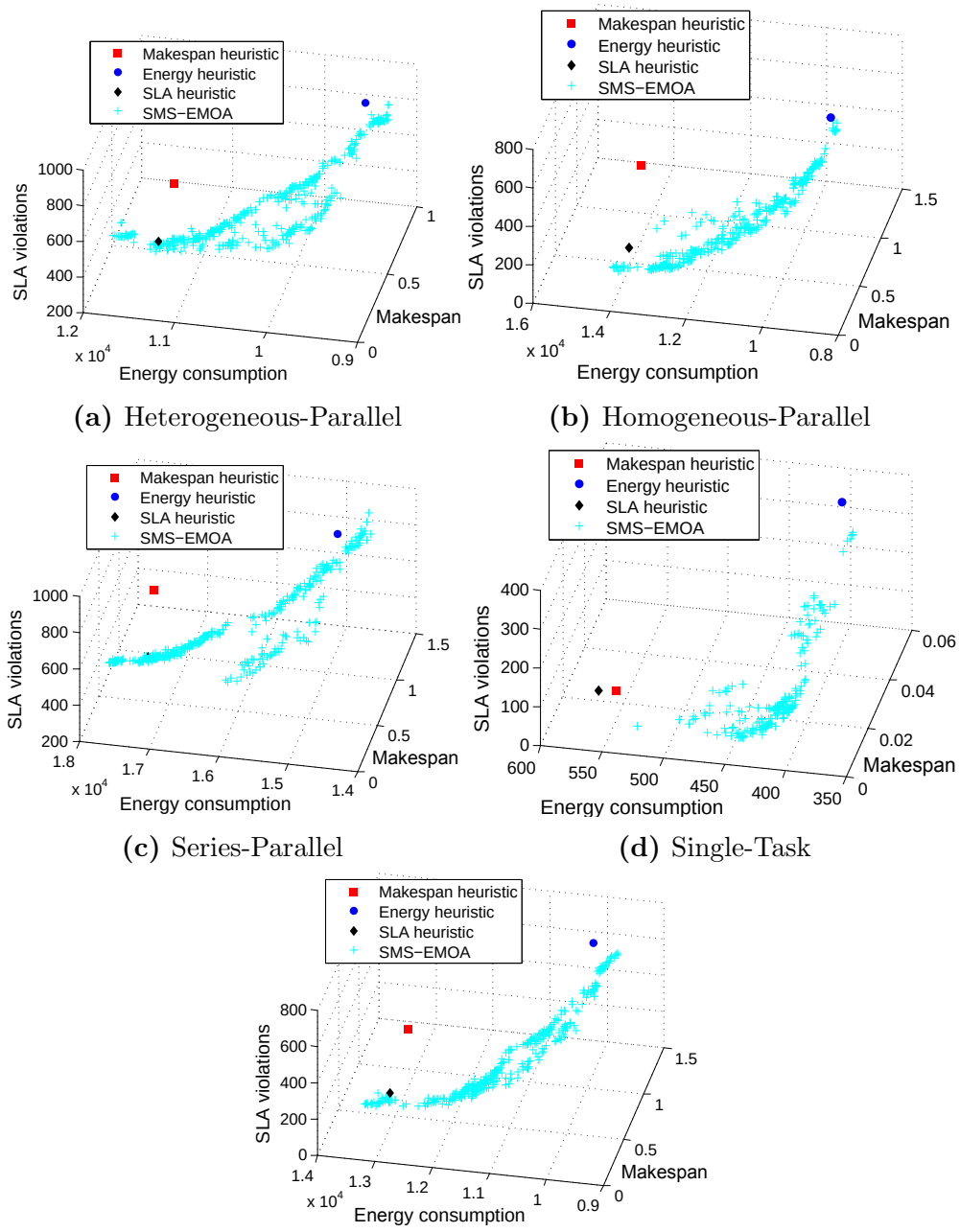
5.5 Conclusions

In this work we propose a multiobjective formulation for modeling the scheduling of a large number of workflows in a federation of datacenters to simultaneously minimize three objectives: makespan, energy consumption, and number of jobs violating a SLA threshold. This formulation extends the formulation proposed in Chapter 4 by considering heterogeneous datacenters and networking communication. This new formulation provides a more realistic modeling for nowadays high performance computing infrastructure.

We consider a two-level hierarchical scheduling approach to address the proposed problem. The high-level algorithm schedules workflows to datacenters, while the low-level algorithm schedules the tasks of the workflows assigned to each datacenter to the servers in that datacenter. This two-level strategy simplifies the scheduling by dividing the whole problem into two sub-problems.

Regarding the high-level scheduler, we propose a total of 56 heuristic algorithms for online scheduling, and 3 MOEAs for offline scheduling. We study and compare the accuracy of all the proposed high-level schedulers considering a set of 100 diverse and realistic problem instances.

The analysis of the proposed high-level heuristics shows CA-MaxMin+U, CA-MaxMin+AC+ASC, CA-MaxMin+MC+ASC, and CA-MaxMin+L+ASC are the most accurate heuristics for makespan optimization. LF+AC+DSC and LF+MC+DSC are the most accurate heuristics for energy consumption optimization, and CA-MaxMin+L+DSC is the most accurate for SLA violations optimization.



(a) Heterogeneous-Parallel

(b) Homogeneous-Parallel

(c) Series-Parallel

(d) Single-Task

(e) Mix workflows

Figure 5.3: Sample results computed by the best heuristics and SMS-EMOA for the large-sized scenarios

The proposed SMS-EMOA method proved to be overall the most accurate MOEA scheduler, significantly outperforming IBEA and NSGA-II on most multiobjective metrics for most of the problem instances. In average, SMS-EMOA is able to improve makespan by 20%, energy consumption by 10% and SLA violations by 16% over the best heuristics for the small-sized scenarios.

For the large-sized scenarios, SMS-EMOA improves makespan by 43%, energy consumption by 2% and SLA violations by 42% in average. Furthermore, it is able to compute a diverse set of trade-off schedules with different levels of compromise between all three objectives.

The main line of future work consists in developing a mixed integer programming solution for computing exact lower-bounds for studying the optimality gap for each objective. Also, we propose to evaluate our proposed method with even larger problem instances to further study the behavior of the computed results.

Chapter 6

An empirical study of the robustness of energy-aware schedulers

Santiago Iturriaga, Sebastián García, and Sergio Nesmachnow

Abstract: This article presents an empirical evaluation of energy-aware schedulers under uncertainties in both the execution time of tasks and the energy consumption of the computing infrastructure. We address an important problem with direct application in current clusters and distributed computing systems, by analyzing how the list scheduling techniques proposed in a previous work behave when considering errors in the execution time estimation of tasks and realistic deviations in the power consumption. The experimental evaluation is performed over realistic workloads and scenarios, and validated by in-situ measurements using a power distribution unit. Results demonstrate that errors in real-world scenarios have a significant impact on the accuracy of the scheduling algorithms. Different online and offline scheduling approaches were evaluated, and online approach showed improvements of up to 32% in computing performance and up to 18% in energy consumption over the offline approach using the same scheduling algorithm.

6.1 Introduction

Nowadays, energy efficiency is a major concern when operating clusters, datacenters, and grid/cloud computing infrastructures. From a global perspective,

all issues related to energy consumption raise several concerns for the scientific community, including economic, environmental, and system performance (Lee and Zomaya, 2009).

Energy consumption on computing systems does not only depend on the energy efficiency and features of the hardware, but also on the software used for task planning (Ahmad and Ranka, 2012). Among many different strategies for reducing the energy consumption,energy-aware scheduling techniques have emerged as useful alternatives for accurate planning and lowering the power required for operation (Nesmachnow et al., 2013). Energy reduction techniques are usually based on limiting the computing power of the computing elements. They are in conflict with the system performance, so applying them has an impact on the Quality of Service (QoS) perceived by the user. Multi-objective formulations of the scheduling problem have been formulated to account for the specific features of the trade-off between energy utilization and performance (Dorronsoro et al., 2010).

The main trend on the scientific community in energy-aware scheduling is based on optimizing the energy consumption of the computing elements since the processor is the main energy consuming element among the hardware components. The processor also offers the most flexible options for energy management, such as dynamic voltage and frequency scaling (DVFS), dynamic power management, slack sharing and reclamation, and other techniques (Zhu et al., 2003).

Many scheduling algorithms are based on assuming that the time required to perform every task is known in advance, and the planning is performed according to that input information. However, that assumption does not hold true in the case of computational infrastructures, where users submit their jobs to be executed on heterogeneous computing elements. Accurately predicting the execution time for individual tasks is a very hard problem, mainly because the actual execution time depends on many factors including the hardware features, communications and delays due to infrastructure and parallel execution, resource availability, among others. Estimation models using task profiling and benchmarking have been proposed since the early 1990's (Ghafoor and Yang, 1993; Kafil and Ahmad, 1998), but they rely on specific hardware features and computing models that are not fully reasonable for nowadays clusters and distributed computing infrastructures. Furthermore, current models for predicting the energy consumption do include some unrealistic approxima-

tions about the power utilization, especially in the case of complex multicore servers (Nesmachnow et al., 2013).

This article presents an empirical evaluation of energy-aware schedulers in heterogeneous computing (HC) scenarios that consider uncertainties in both the execution time of tasks and the energy consumption for a given computing infrastructure. We propose three variants of each of the best energy-aware list scheduling techniques proposed in our previous work (Nesmachnow et al., 2013). Then, we analyze their behavior when addressing specific instances of the energy-aware scheduling problem in multicore HC systems, accounting for realistic errors in the estimation of the execution time of tasks, and specific deviations in the power consumption calculation when using a standard energy model for computing systems.

The main contribution of this article consists in proposing novel scheduling algorithms and reporting their experimental evaluation performed over realistic workloads and scenarios, validated by in-situ measurements using a power distribution unit. The empirical results demonstrate that error in real-world scenarios have a significant impact on the accuracy of the scheduling algorithms. Different scheduling approaches were evaluated, and the online approach showed improvements of up to 32% in computing performance and up to 18% in energy consumption over the offline approach using the same scheduling algorithm.

The paper is organized as follows. Section 6.2 describes the energy-aware scheduling problem under uncertainty. A review of related work is presented in Section 6.3. The heuristics for energy-aware scheduling in high performance computing systems are introduced in Section 6.4, just before the description of our model for uncertainty in Section 6.5. The experimental analysis of the proposed heuristics is reported in Section 6.6, Finally, Section 6.7 presents the conclusions and formulates the main lines for future work.

6.2 Robust energy-aware scheduling under uncertainty

This section describes the robust energy-aware scheduling problem in HC systems under conditions of uncertainty.

6.2.1 The energy-aware scheduling problem

In this article, we consider a multi-objective version of the scheduling problem in multicore HC systems, taking into account the minimization of the makespan and energy consumption. We call this problem the Makespan-Energy Heterogeneous Computing Scheduling Problem (ME-HCSP). The mathematical formulation for the problem considers the following elements:

- A HC system composed of a set of multicore machines $P = \{m_1, \ldots, m_M\}$; each machine having $NC(m_i)$ processing cores and processing speed $S(m_i)$.
- A collection of tasks $T = \{t_1, \ldots, t_N\}$ to be executed on the system, each task arrives in time $ARR(t_i)$.
- An *execution time function* $ET : T \times P \to \mathbf{R}^+$, where $ET(t_i, m_j)$ is the time required to execute task t_i on machine m_j.
- An *execution time error function* $\Delta_{ET} : T \times P \to \mathbf{R}^+$, where $\Delta_{ET}(t_i, m_j)$ is the error introduced when estimating $ET(t_i, m_j)$.
- An *energy consumption function* $EC : T \times P \to \mathbf{R}^+$, where $EC(t_i, m_j)$ is the energy required to execute task t_i on machine m_j, and $EC_{IDLE}(m_j)$ is the energy that machine m_j consumes in idle state.
- An *energy consumption error function* $\Delta_{EC} : T \times P \to \mathbf{R}$, where $\Delta_{EC}(t_i, m_j)$ is the error introduced when estimating $EC(t_i, m_j)$.

The goal of the ME-HCSP is to find an assignment function $f : T^N \to P^M$ which simultaneously minimizes the *makespan* and the *total energy consumption* metrics. The assignment function f should schedule each task t_i to be executed without preemption on some machine m_j at some time $ST(t_i)$, with $ST(t_i) \geq ARR(t_i)$. The makespan metric is defined as the maximum completion time $C_{max} = \max_{t_i \in T} C(t_i)$, where the completion time of task t_i is $C(t_i) = ST(t_i) + ET(t_i, m_j)$. The energy required to execute the task t_i in the machine m_j, given by $EC(t_i, m_j)$, depends on the execution time of the task t_i in machine m_j, $ET(t_i, m_j)$, and the energy consumption of the machine m_j. The *total energy consumption* is defined as shown in Equation 6.1.

$$\sum_{\substack{t_i \in T: \\ f(t_i) = m_j}} EC(t_i, m_j) + \Delta_{EC}(t_i, m_j) + \sum_{m_j \in P} EC_{IDLE}(m_j) \tag{6.1}$$

Regarding the energy consumption, in this work we apply the model for multicore computing systems introduced in our previous work (Nesmachnow et al., 2013). In this model, the energy consumption of a task is estimated by assuming the task is CPU-bound and approximating its energy consumption by the energy consumption of the CPU when executing that task. This was found to be an accurate approximation in an HPC systems where most tasks are CPU intensive and where the CPU is the most energy consuming device. This model states that the total energy consumption accounts for both the energy required to execute the assigned tasks, and the energy that each machine consumes in idle state. Therefore, we do estimations for the worst case scenarios because in real systems idle machines can be changed to an energy saving mode (or switched off).

In the previous formulation all tasks can be independently executed, disregarding the execution order. The independent task model is common in grid and volunteer-based computing infrastructures, as well as in BoT applications.

6.2.2 Robust scheduling

Most modern High Performance Computing (HPC) systems are comprised of a large number of distributed and heterogeneous computing elements. The execution times of tasks in these HPC systems is inherently unpredictable (Cirne and Berman, 2001; Mu'alem and Feitelson, 2001); computing element heterogeneity and network communication delays contribute a great deal to task execution time uncertainty. But arguably the major factor of uncertainty when scheduling tasks in HPC systems is introduced by the users of the system when specifying the *Estimated Execution Time* (EET), defined as $EET = ET + \Delta_{ET}$. The ET of all tasks is a very important component in order to compute an accurate task schedule, but because it is unknown for the scheduling algorithm, nowadays HPC systems relay on user estimates of tasks execution times, the EET. This is true for most of the modern scheduling products such as Load Leveler, Maui, Open Grid Scheduler, etc. (Tsafrir, 2010).

Studies show the EET estimates are highly inaccurate, in some cases the ET of a significant number of jobs account for 10% or less of their EET (Bailey Lee et al., 2005). There are a number of reasons for the high inaccuracy of the EET estimates. The first being that a significant number of tasks fail to execute because of task initialization errors. Though this is more related to

configuring errors than to inaccurate EET, it still needs be considered by the scheduling algorithm. Another reason is that tasks that do execute correctly are largely overestimated. This is because many systems kill an executing task after its EET has been consumed, hence the EET estimate is not the true user estimate, but rather the maximum amount of time the user is willing to wait for the task execution output before it is acceptable for the task to be killed by the system (Mu'alem and Feitelson, 2001). Real-world execution traces show this is true even for tasks following the independent task model (Tsafrir, 2010).

Energy consumption estimation is greatly affected by execution time estimation errors since energy consumption directly depends on the execution time of the scheduled tasks. But this is not the only uncertainty source; although the CPU is the most energy consuming device in HPC systems, certainly it is not the only one. Energy consumption is also affected by the use of peripheral devices (such as hard drives, network adapters, etc.) and by the use of cooling devices (such as cooling fans, air conditioning, etc.).

Uncertainty in the energy consumption and the execution time of tasks in HPC systems can lead to a considerable performance loss in task execution (Tsafrir, 2010). Hence, looking for scheduling solutions that are robust against such inaccuracies may help alleviate, or even neglect, the performance decrease they produce.

6.3 Related Work

Several works in the related literature have studied algorithms to find *flexible* solutions to the scheduling problem, i.e. they are able to handle some kind of uncertainties related to faults in the system, or they are expected to be less affected by these uncertainties than a regular scheduler.

Ali et al. (2004) proposed a mathematical formulation of a metric for the robustness which can be applied to various parallel and distributed systems. The authors apply this metric to two example systems, one of them being the independent application allocation system that we are considering in this paper. When adopting this robustness metric, it is guaranteed that if the collective difference of the actual task execution times versus the estimated times is within a certain calculated range, then the given makespan requirement will be met. This metric has been used in a number of related works (Dorronsoro et al., 2010; Mehta et al., 2007).

Several works aim at predicting an uncertainty value in order to include this prediction into the scheduling knowledge. All these works focus on predicting execution time uncertainty while considering a simple FCFS scheduling approach; they do not consider the energy consumption of the system. Tsafrir (2010) proposed a system-generated prediction system based on users' history and applied it to the EASY (Feitelson et al., 2005) algorithm. Using this approach they achieved a 25% average reduction in wait time and slowdown. Minh and Wolters (2010) presented a method for predicting task execution time based on historical data. Using this predictor they were able to improve accuracy by up to 32%. The CREASY scheduler by Shmueli and Feitelson (2009) exploits knowledge on user behavior to improve QoS of the system. Using an alternative simulation methodology called site-level simulation they were able to improve user productivity by up to 50%. Tang et al. (2013) analyzed the impact of execution time estimates in scheduling algorithms on the Blue Gene/P, designing and implementing a number of schemes for adjusting estimates. These schemes make use of historical workload data in order to predict the accuracy of a task estimation considering user and project information. The analysis showed the user estimates are highly inaccurate with only 31–33% of all the considered tasks having an estimation accuracy of 80% or more, and up to 21–28% having an accuracy of 20% or less. The experiments showed the adjusting schemes were able to improve up to 20% the performance of the system.

In our previous work (Nesmachnow et al., 2013), the model for multi-core computing systems that we apply in this article was introduced. Our approach did not apply DVFS nor other specific techniques for power/energy management. Instead, we proposed an energy consumption model (MIN/MAX model) based on the energy required to execute tasks at full capacity (E_{MAX}), the energy when not all the available cores of the machine are used, and the energy that each machine on the system consumes in idle state (E_{IDLE}). In our previous work, we proposed twenty fast list scheduling methods adapted to solve the bi-objective problem we also consider here, by simultaneously optimizing both makespan and energy consumption when executing independent BoT applications on a computing system composed of multi-core computers.

In this work we propose to study the impact on real-world scenarios of both execution time and energy consumption uncertainties when considering system performance and energy efficiency objectives. We evaluate a set of

online and offline variants of scheduling algorithms proposed by Nesmachnow et al. (2013) which simultaneously consider both objectives. To the best of our knowledge, this is the first work to evaluate energy consumption uncertainty in a computing scheduling problem.

6.4 Robustness of energy aware scheduling heuristics

In this work we consider three well-known scheduling approaches which can be classified as *offline*, *online greedy*, and *online batch* (Leung et al., 2004). The offline approach assumes all tasks are known beforehand, hence the scheduling algorithm needs only to be executed once and is able to consider all the tasks simultaneously for the scheduling decisions. This approach is definitely the best in an uncertainty free problem, since the scheduling algorithm is provided with absolutely all the available information for making scheduling decision. Unfortunately, because the scheduling algorithm is executed only once, it is unable to dynamically adjust the scheduling to cope with uncertainty values.

On the other hand, in the online approach tasks are not known by the scheduling algorithm until they arrive. This requires the scheduling algorithm to be executed multiple times for completely scheduling a task workload. We tackle the online scheduling problems using two different techniques, one is a greedy technique and the other is a batch oriented technique. In the online greedy approach, tasks are scheduled one at a time as soon as they arrive and are never rescheduled. This approach is very simple and straightforward, and is able to react to some degree to uncertainty in the data. On the downside, the information available to the scheduler for making the scheduling decisions is minimal.

The online batch approach tackles some of the previously presented problems. In this approach the scheduling algorithm is re-executed after a predefined time step, all the tasks that arrive in a given time-step are delayed, are grouped as a batch, and are scheduled together by the scheduling algorithm. This way the online problem is treated as a succession of smaller offline problems. We consider two further improvements to this approach. The first being that in every scheduling batch not only the tasks that arrive in that time step are considered by the scheduler, but also all the tasks from previous

batches already scheduled but which have not started their execution (i.e. are still queued). The second improvement is that the scheduling algorithm is not executed in every time step, it is executed only if in the current time step some meaningful event has occurred (i.e. at least one task has finished or at least a new task has arrived).

In this work we evaluate five different scheduling algorithms following the previous approaches. For the offline and online batch approaches we considered three multiobjective two-phase list-scheduling algorithms proposed by Nesmachnow et al. (2013): MaxMin, MaxMIN, and SuffMIN. Because a two-phase approach is not applicable to the online greedy approach, two simple single-objective algorithms were proposed: Min and MIN. The algorithms work as follows:

- *MaxMin* is a traditional two-phase heuristic which considers the makespan objective in both phases. In the first phase the task t with the largest compute time is selected. In the second phase task t is assigned to the machine which minimizes the makespan.
- *MaxMIN* is a two-phase heuristic which considers the makespan objective in the first phase and the energy consumption in the second. In the first phase the task t with the largest compute time is selected. In the second phase task t is assigned to the machine which minimizes the energy consumption.
- *SuffMIN* again considers the makespan objective in the first phase and the energy consumption in the second. In the first phase the task t which suffers the most if not assigned right away is selected. In the second phase task t is assigned to the machine which minimizes the energy consumption.
- *Min* and *MIN* are one-phase greedy heuristics that assign tasks as they arrive, considering makespan (*Min*) and energy consumption (*MIN*).

6.5 Modeling uncertainty

In this work we consider two sources of uncertainty, the task execution time (ET) and the machine energy consumption (EC). We present here the task execution time model and the energy consumption model proposed in this work.

6.5.1 The task execution time uncertainty model.

One of the most popular models for modelling execution time uncertainty is the f-model (Mu'alem and Feitelson, 2001). This model assumes the task's EET is uniformly distributed within $[ET, (f+1)ET]$, where f is some positive factor. When $f = 0$ then $\Delta_{ET} = 0$ hence estimates are identical to execution times, and the larger the f-value the greater the user inaccuracy in the system. In this work we perform some empirical analysis and show the f-model does not fit the empirical data considered in the analysis, hence we deduce some simple model from the data in order to model task execution time uncertainty in this work.

In order to construct a model for uncertainty in the tasks execution time we performed an empirical study using workloads from three real-world HPC infrastructures. The analysis is two-fold, first we studied the EET of the tasks to characterize the user behavior when requesting execution time for their tasks, and second we studied the Δ_{ET} of the tasks considering their requested EET.

The first analyzed infrastructure is the CEA Curie system, a large HPC infrastructure with 93312 cores during the considered time span. A workload with 773138 tasks, which spans for 20 months (Feb. 2011–Oct. 2012), was used. We also studied the RICC infrastructure, a medium sized system with 9216 cores. A workload with 447794 tasks, which spans for 5 months (May 2010 to Sept. 2010) was used. Finally, we studied the Cluster FING system, a small sized system which was comprised of 408 cores during the considered time span. For the Cluster FING system a 31 months period was analyzed, in this period dated between November 2011 and June 2014, a total of 500000 tasks were executed.

The CEA Curie and RICC task workloads are available at the Parallel Workloads Archive `http://www.cs.huji.ac.il/labs/parallel/workload` while the Cluster FING workload is available at `www.fing.edu.uy/cluster`.

When analyzing the EET of the tasks in the studied real-world workloads, we found that most EETs are within either less than 20% or more than 80% of the maximum execution time allowed in the system. Hence, we propose grouping tasks in the workloads in 5 different groups: in the first group tasks with EET between 0% and 20% of the maximum execution time, in the second group tasks with EET between 20% and 40%, and so on (see Fig. 6.1).

relative EET	*percentage of tasks*
$[0, 20\%)$	50%
$[20\%, 40\%)$	2%
$[40\%, 60\%)$	2%
$[60\%, 80\%)$	1%
$[80\%, 100\%]$	45%

relative EET error	*percentage of tasks*		
	Δ_{ET}^{low}	Δ_{ET}^{med}	Δ_{ET}^{high}
$[0, 40\%)$	45%	35%	25%
$[40\%, 75\%)$	30%	25%	15%
$[75\%, 95\%)$	5%	10%	20%
$[95\%, 100\%]$	20%	30%	40%

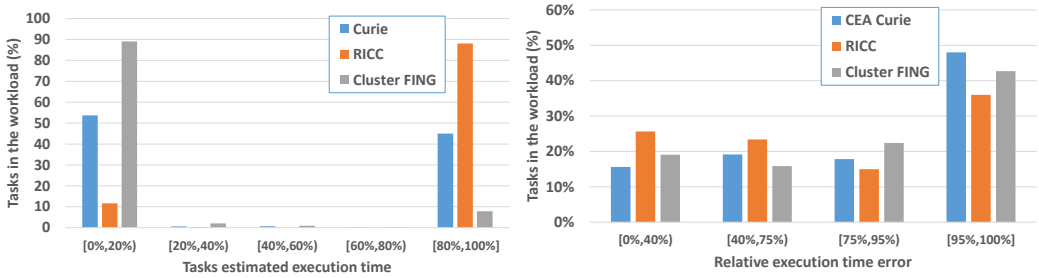


Figure 6.1: Analysis of the proposed workloads.

When averaging the results for the three real-world workloads, we see that in average 50% of the tasks request less than 20% of the maximum allowed execution time, 45% of the tasks request more than 80% of the maximum execution time, and the remaining 5% is somewhat uniformly distributed.

Regarding Δ_{ET}, the workload analysis showed that the estimation errors are rather large and, again, not uniformly distributed. Further analysis showed that a significantly large number of tasks present either a quite accurate estimation or a very inaccurate estimation. This is shown in Fig. 6.1. This empirical findings are similar to the ones presented by Tsafrir (2010). Based on this data we propose three different error scenarios for our model: Δ_{ET}^{low} with an average error of 48%, Δ_{ET}^{med} with an average error of 56%, and Δ_{ET}^{high} with an average error of 67%.

6.5.2 The energy consumption uncertainty model.

We conducted a set of empirical evaluations in order to determine the uncertainty model for the energy consumption.

Our starting point was the high-level theoretical linear increasing model that we originally introduced in our previous work (Nesmachnow et al., 2013). This model proposes a linear increase in the energy consumption (from E_{IDLE} to E_{MAX}) when using an increasing number of CPU cores. However, the model is only focused on the energy consumption of the processor; it does

not take into account the energy consumption due to memory utilization and I/O devices. Thus, in this work we aim at validating the energy consumption model and estimate deviations from the previous model due to other energy consuming components.

In order to evaluate the model, three basic tests were executed using a server from our HPC infrastructure at Universidad de la República. The server is an HP Proliant DL385 G7 server with two AMD Opteron 6172 processors with 12 cores running at 2.1 GH, and 24 GB of RAM memory.

For the energy evaluation, a specialized Power Distribution Unit (PDU) was used: CyberPower PDU20SWHVIEC8FNET. We connected only the server running the tests to the PDU, as it lacks the capability of per outlet measurement. A specific application was developed to poll and log the energy consumption data, due to a limitation on the granularity of the logging capabilities of the PDU, which is only able to save log data at a rate of one measurement per minute. The logging application was executed in a separated computer also connected to the PDU, in order to avoid adding its own energy consumption to the measurements. Using the logging application, we were able to log a minimum of four and a mean of six instant energy measurements per second during each test.

The tests consist in executing an increasing number of applications in order to use different number of cores, from a single core up to twenty four cores. The applications used in the tests range from a simple mathematical operation to a complex transformation, in order to evaluate different scenarios:

1. *Single loop.* The first test consists on running a simple C++ loop performing a multiplication a huge number of times, this way ensuring a fully CPU-bound test using only one CPU.
2. *LINPACK.* The second test is based on an open source sequential implementation of the LINPACK benchmark (Dongarra, 1988). We adjusted the LINPACK parameters to have an acceptable execution time while not using too much memory, to reduce the race for cache and RAM memories when running 24 instances.
3. *Fast Fourier Transform.* This test is similar to the previous one, but based on an open source implementation of the Fast Fourier Transform (Brigham, 1974). In this case, the evaluation was made using only up to 23 instances of the test.

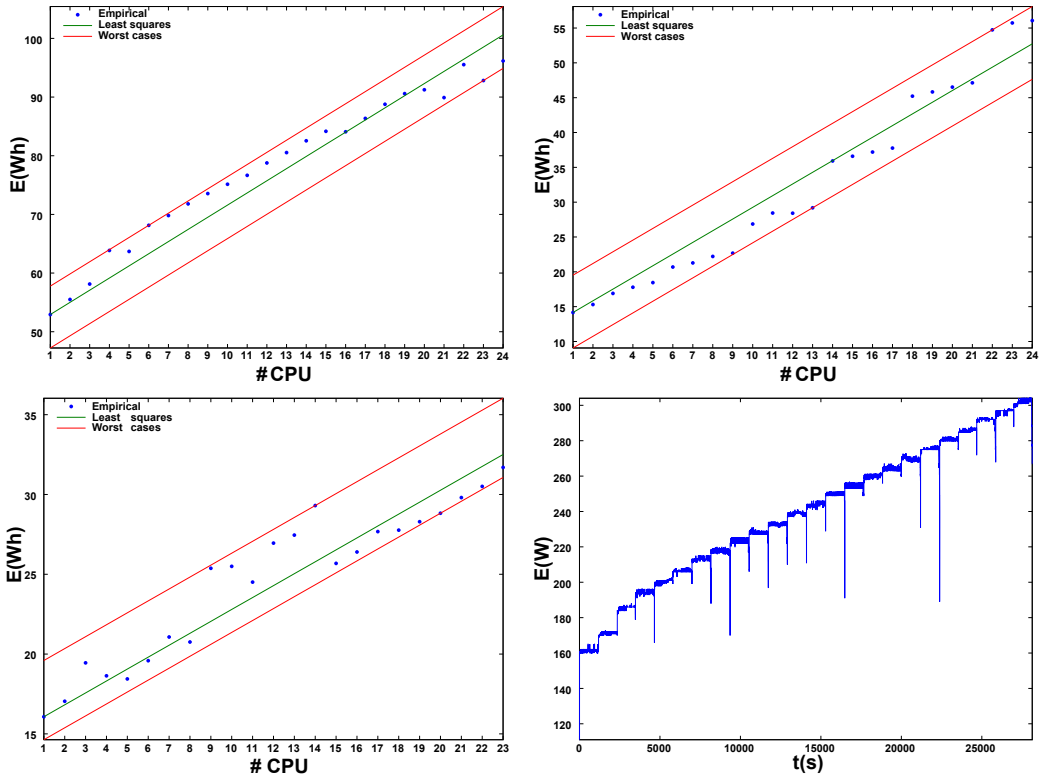


Figure 6.2: Energy consumption for the three applications in the test (loop, LINPACK, and FFT, respectively), and instant power usage sample (loop test case)

Table 6.1: Error results and deviation from linearity for the three tests performed

test	*error*		*deviation from linearity*		*relative error*	
	maximum	*mean*	*maximum*	*mean*	*maximum (%)*	*mean (%)*
simple loop	5.71	2.56	2.48	1.11	7.36	**3.34**
LINPACK	5.36	2.39	2.74	1.23	21.58	**7.98**
FFT	3.52	1.26	2.82	1.01	13.11	**4.88**

In the tests, the energy consumption was estimated from the logs obtained using the PDU by applying an interpolation of the instant power measurements. The graphics in Figure 6.2 shows the energy consumption when using an increasing number of cores for the three applications in the test (loop, LINPACK, and FFT, respectively). The fourth graphic in Figure 6.2 is an example of the instant power usage as function of time for the loop test case, where the execution of the tests using an increasing number of cores were performed one after the other.

Table 6.1 reports the maximum (worst case) and mean values for the error and the deviation from linearity in the energy consumption, along with the relative error values for each application in the test suite.

The numerical results validates the linear increasing energy consumption model, as we verify that the deviation from linearity when using real applications is below 3%; and the relative error on the energy consumption is below 8%, and about 5% in average. These results demonstrate that no significant impact is observed when executing CPU-oriented applications, such as the ones commonly executed in HPC facilities. Taking into account the results of the empirical analysis, we assume that the energy consumption error for multicore computers is in the range [-5%,5%].

6.6 Experimental analysis

This section reports the experimental analysis of the proposed heuristics for robust energy-aware scheduling under uncertainty.

6.6.1 Problem instances

We created a number of problem instances to evaluate the scheduling algorithm using the proposed uncertainty model. Each problem instance is defined by the task *workload*, describing the tasks to be executed in the system, and the machine *scenario*, describing the hardware infrastructure to execute the tasks.

The machine scenarios were created using the model for energy consumption in multicore computers (Nesmachnow et al., 2013) which makes use of a list of CPU and generates each scenario selecting machines using a uniform probability distribution. However, in this work we propose an alternative machine selection method for constructing each scenario: the CPUs are sorted according to their generation, the mean of the Gaussian probability distribution is uniformly selected, and two different standard deviation values are used, σ_{high} and σ_{low}. These σ values represent the machine heterogeneity in the generated scenario and they are defined as σ_{high}=0.25×M and σ_{low}=0.025×M, where M is the number of machines in the scenario. This method models a more realistic computing infrastructure comprised of sets of similar machines.

Scenarios of three different sizes were generated for this work following this new approach, $M \in \{8, 12, 16\}$. A total of **800** scenarios were generated for each of the considered number of machines, with the 8-machine scenarios comprising an average of 131 cores per scenario, and the 16-machine scenarios comprising an average of 262 cores per scenario.

Regarding the task workload generation, 1024 tasks were generated for each workload using a Poisson probability distribution to model their arrival time. The experiments were performed using the lowest and highest average arrival rates of the three real-world workloads analyzed, $\lambda_{low} = 0.317$ and $\lambda_{high} = 0.634$. With this settings, the average simulation time of each 1024-tasks workload is around 53 minutes when using λ_{low} and around 26 minutes when using λ_{high}.

We fixed the maximum allowed time for each task execution to be 28 hours, which considering the proposed uncertainty model results in an average task EET of 13.7 hours and an average task ET of 7.8 hours.

A total of **400** task workloads were generated, 50 workloads for each combination of execution time error rate (Δ_{ET}) and arrival rate (λ). Each workload is evaluated with two machine scenarios, with high and low heterogeneity. Hence, a total of 800 experiments were conducted with different problem instances.

6.6.2 Results and discussion

In this section we present and discuss the experimental analysis results for all the performed experiments.

First we explore the deviation from the expected schedule when using the offline scheduling algorithms. Table 6.2 presents the relative deviation between the expected and the actual makespan and energy consumption for each algorithm. Because of the nature of the problem the expected makespan and energy consumption is an upper bound of the actual values of the schedule, hence all deviation is an improvement form the expected schedule.

We can see the schedule deviation in both objectives increases as the error rate increases, and decreases as the problem dimension increases. When comparing the scheduling algorithms, results show the MaxMIN algorithm is the most robust for the makespan objective, while SuffMIN is the most robust for the energy consumption objective. However, the gap between the expected and the actual metrics of the schedules is significant for all the scenarios and all the scheduling algorithms. The best results are marked in **bold**.

Table 6.3 compares the considered algorithms showing their average relative improvement compared to the worse performing algorithm for each scenario and objective. Results show the offline MaxMin computes the most accurate schedules for both objectives in every scenario when the error rate (Δ_{ET})

Table 6.2: Average makespan and energy deviation for the offline algorithms

	num. machines	Δ_{ET}	offline heuristic		
			MaxMin	**MaxMIN**	**SuffMIN**
makespan	8	low	24.2%	**16.8%**	29.9%
		med.	31.4%	**22.2%**	37.9%
		high	38.4%	**28.0%**	46.6%
	12	low	22.1%	**13.1%**	28.9%
		med.	27.7%	**16.7%**	36.0%
		high	34.0%	**21.7%**	45.0%
	16	low	20.7%	**11.7%**	27.8%
		med.	23.6%	**13.1%**	33.3%
		high	29.2%	**18.1%**	42.0%
energy	8	low	33.7%	37.6%	**26.8%**
		med.	42.0%	45.5%	**35.9%**
		high	53.1%	54.8%	**48.5%**
	12	low	30.6%	35.7%	**15.5%**
		med.	38.1%	42.9%	**26.5%**
		high	50.1%	52.5%	**41.9%**
	16	low	28.5%	34.8%	**11.3%**
		med.	37.0%	41.5%	**20.0%**
		high	46.5%	50.5%	**32.3%**

is none, increasing its accuracy as the problem dimension increases. This was expected as the offline algorithm is the one considering the greater amount of scheduling information. When considering problem scenarios with higher error rates, it can be seen that the online batch algorithms outperform the offline algorithms. The scheduling algorithms using the online approach are able to react to uncertainty and improve the accuracy of the schedule. The online batch MaxMin computes the most accurate schedules for the makespan objective, and the online batch MaxMIN computes the most accurate schedules for the energy consumption objective. It can be seen that the accuracy of the online batch algorithms increases with the problem dimension and the error rate, achieving an improvement of up to 50.2% for the makespan and up to 32.0% for the energy consumption. When comparing the online batch and offline approaches of the best algorithms, results show the online batch MaxMin is up to 32.3% more accurate than the offline MaxMin for the makespan, and the online batch MaxMIN is up to 18% more accurate than the offline MaxMIN for the energy consumption.

Table 6.3: Average makespan and energy consumption improvement

		Δ_{ET}	heuristic							
			offline			online				
						batch			greedy	
			MaxMin	MaxMIN	SuffMIN	MaxMin	MaxMIN	SuffMIN	Min	MIN
makespan	*8 mach.*	none	**34.7**%	22.0%	0.7%	34.4%	34.0%	29.8%	27.4%	28.0%
		low	23.1%	15.8%	2.4%	**44.4**%	44.2%	38.1%	22.7%	21.8%
		med.	22.0%	16.2%	3.4%	**47.2**%	46.3%	40.6%	22.2%	19.9%
		high	16.8%	15.0%	5.4%	**49.1**%	48.4%	43.2%	20.0%	19.3%
	12 mach.	none	**43.6**%	29.1%	1.3%	42.7%	40.9%	36.3%	35.1%	35.4%
		low	31.0%	23.3%	1.5%	**48.3**%	47.5%	42.1%	28.7%	28.0%
		med.	26.8%	19.9%	2.1%	**49.5**%	47.8%	43.2%	26.4%	25.3%
		high	21.7%	18.6%	3.5%	**48.6**%	47.9%	43.0%	24.1%	22.2%
	16 mach.	none	**48.4**%	32.9%	1.7%	47.0%	43.3%	39.3%	39.1%	38.9%
		low	33.2%	24.5%	1.2%	**47.9**%	46.0%	40.7%	28.9%	28.1%
		med.	32.0%	25.5%	1.2%	**50.2**%	48.5%	43.4%	29.6%	29.1%
		high	28.4%	22.7%	2.4%	**49.6**%	48.4%	44.5%	28.1%	27.0%
energy	*8 mach.*	none	**13.4**%	8.9%	0.6%	13.2%	13.1%	11.6%	10.8%	11.0%
		low	10.7%	8.0%	1.9%	21.7%	**22.1**%	19.1%	10.8%	10.2%
		med.	10.6%	8.8%	2.7%	24.4%	**24.5**%	21.4%	11.1%	9.7%
		high	9.0%	9.2%	4.2%	27.5%	**27.8**%	24.9%	11.2%	10.6%
	12 mach.	none	**19.4**%	13.9%	0.8%	19.0%	18.4%	16.5%	15.7%	15.9%
		low	15.8%	13.1%	1.2%	25.8%	**26.3**%	23.3%	15.1%	14.6%
		med.	14.4%	11.9%	1.9%	**27.9**%	27.8%	25.0%	14.5%	13.8%
		high	12.2%	11.6%	2.8%	28.7%	**29.2**%	26.2%	14.5%	13.2%
	16 mach.	none	**23.6**%	17.5%	1.1%	22.9%	21.6%	19.5%	19.5%	19.4%
		low	17.4%	14.2%	1.0%	**26.2**%	26.1%	23.0%	15.5%	14.9%
		med.	17.9%	15.7%	1.0%	**29.4**%	29.4%	26.1%	17.1%	16.8%
		high	17.6%	15.3%	2.1%	31.6%	**32.0**%	29.0%	18.1%	17.3%

Table 6.4 shows the number of problem instances in which each algorithm is able to compute the most accurate schedule for each objective. It can be seen that the previous results hold. The most accurate heuristic is the offline MaxMin when no error level is considered. The online batch MaxMin is the most accurate for the makespan objective when higher error rates are considered, and the online batch MaxMIN is the most accurate for the energy consumption objective also when higher error rates are considered. Although the online greedy algorithms are able to compute competitive schedules in average, they are not able to compute the most accurate result for any problem instance.

6.7 Conclusions and Future Work

This work presented a formulation for the energy-aware scheduling problem considering uncertainties in the execution time of tasks and the energy consumption of the infrastructure. We analysed three real-world task workloads and proposed a workload generation model considering uncertainties.

Table 6.4: Number of problem instances in which each of the proposed heuristic compute the best makespan and energy consumption value

		Δ_{ET}	heuristic							
			offline			online				
						batch			greedy	
			MaxMin	**MaxMIN**	**SuffMIN**	**MaxMin**	**MaxMIN**	**SuffMIN**	**Min**	**MIN**
makespan	*8 mach.*	none	**153**	2	0	32	13	0	0	0
		low	1	0	0	**108**	89	4	0	0
		med.	2	0	0	**125**	70	5	0	0
		high	1	0	0	**102**	82	16	0	0
	12 mach.	none	**151**	1	0	24	23	1	0	0
		low	7	0	0	**115**	74	4	0	0
		med.	7	2	0	**122**	61	12	0	0
		high	4	0	0	**91**	89	16	0	1
	16 mach.	none	**170**	1	0	20	9	0	0	0
		low	18	0	0	**111**	69	2	0	0
		med.	11	0	0	**101**	85	7	0	0
		high	14	1	0	**96**	78	13	0	0
energy	*8 mach.*	none	47	13	2	**69**	58	11	0	0
		low	0	0	0	73	**117**	10	0	0
		med.	0	2	1	91	**100**	6	0	0
		high	0	0	0	77	**103**	20	0	0
	12 mach.	none	**80**	10	0	53	53	4	0	0
		low	5	0	0	66	**119**	10	0	0
		med.	2	2	0	85	**94**	16	0	1
		high	1	0	0	63	**108**	26	1	1
	16 mach.	none	**93**	22	3	33	46	3	0	0
		low	10	3	0	76	**101**	10	0	0
		med.	4	0	0	70	**114**	11	1	0
		high	7	4	0	62	**107**	19	0	1

We also conducted empirical evaluations to validate and extend our previously proposed energy consumption model to consider uncertainty values.

In order to analyse the impact of these uncertainty values we evaluated a set of scheduling algorithms considering different scheduling approaches. Some of these scheduling approaches being better fitted to cope with uncertainties than others. Results show the uncertainty values in real-world scenarios significantly affects the accuracy of the scheduling algorithm, hence considering these uncertainty values may improve the accuracy of a scheduling algorithm.

In future work, we propose to extend our mathematical model to consider parallel non-independent tasks and to characterize the energy consumption of tasks which are not entirely CPU-bound, allowing us to model even more realistic problem instances and to take advantage of technologies such as DVFS. We will work on improving the accuracy of our proposed scheduling algorithms and compare them with some well-known commercial batch scheduler, e.g. Maui.

Chapter 7

Conclusions and future work

This chapter presents a summary of the major conclusions presented in this thesis and outlines the main lines of future work.

7.1 Summary of conclusions

This thesis addresses the challenge of scheduling the operation of energy-efficient data centers. In Chapter 2, we provide an in-depth review of the state-of-the-art knowledge related to energy efficiency for data centers. After that, in Chapters 3 to 6, we address the energy efficiency problem for different data center scenarios.

In Chapter 3 we address the problem of optimizing energy efficiency in a data center powered by traditional and renewable energy sources. This problem takes into account a desired reference power consumption profile, the overall electricity budget, and the QoS provided to its users. We consider a fully Pareto-oriented methodology and designed two MOEAs based on NSGA-II and ev-MOEA. Both MOEAs are hybridized with a greedy scheduling algorithm and a SA algorithm. In our proposed approach, MOEAs schedule the computing resources and cooling system, while the greedy heuristic schedules the computing workload. Furthermore, the SA algorithm is used as a post hoc optimization mechanism for further improving the computed schedules. A diverse set of problem instances are constructed considering different power profiles, green power generation profiles, and workloads of tasks. Results show ev-MOGA is significantly more accurate than NSGA-II for all the considered problem instances, improving accuracy of the schedules by 18% in average.

Compared to the business-as-usual scenario, ev-MOGA computed average budget reductions ranging from 33% up to 83% maintaining high QoS and low deviation from the reference power profile. These results confirm the effectiveness of our proposed approach and the usefulness of considering renewable energy sources for data center infrastructures.

In Chapter 4 we address the multi-objective problem of scheduling a large number of workflows in a federation of several geographically-distributed data centers, each comprised of homogeneous computing resources. The goal of this problem is to simultaneously minimize three objectives: makespan, energy consumption, and number of workflows violating SLA. We propose a two-level hierarchical scheduling algorithm, and we design five heuristics to tackle the online version of the problem and two MOEAs to tackle the offline version of the problem. Both MaxMin-based algorithms are able to compute the most accurate schedules for medium- and large-sized instances when considering makespan and energy consumption objectives. Nevertheless, these algorithms are also the less accurate for SLA violations. The MinMIN-EFT heuristic is the most accurate scheduler for minimizing SLA violations, outperforming MaxMin by 13.5% for large-sized instances. When analyzing the MOEA, results show that Heterogeneous-Parallel is one of the hardest instance classes to tackle, but also one of the most promising. For the medium-sized Heterogeneous-Parallel instances, the MOEAs compute solutions with an average makespan gap of 40.0% and energy consumption gap of 39.3%, improving the best heuristic by 17.9% in makespan and 96.4% in energy consumption. For the large-sized instances, experimental results confirm that Heterogeneous-Parallel is one of the hardest type of instances, with the MOEA computing improvements of 24.2% and 56.9% in makespan and energy consumption over the best heuristic methods.

In Chapter 5 we extend the work presented in Chapter 4 by considering data centers comprised of heterogeneous computing resources and by considering networking communication. This new formulation provides a more realistic modeling for nowadays data center infrastructures. We also extend the experimental analysis by considering a total of 56 heuristic algorithms for online scheduling and 3 MOEAs for offline scheduling. The analysis of the proposed high-level heuristics shows CA-MaxMin+U, CA-MaxMin+AC+ASC, CA-MaxMin+MC+ASC, and CA-MaxMin+L+ASC are the most accurate heuristics for makespan optimization. LF+AC+DSC and LF+MC+DSC are

the most accurate heuristics for energy consumption optimization, and CA-MaxMin+L+DSC is the most accurate for SLA violations optimization. The proposed SMS-EMOA method proved to be the overall most accurate MOEA scheduler, significantly outperforming IBEA and NSGA-II on most multi-objective metrics for most of the problem instances. In average, SMS-EMOA is able to improve makespan by 20%, energy consumption by 10% and SLA violations by 16% over the best heuristics for the small-sized scenarios. For the large-sized scenarios, SMS-EMOA improves makespan by 43%, energy consumption by 2% and SLA violations by 42% in average. Furthermore, it is able to compute a diverse set of trade-off schedules with different levels of compromise between all three objectives.

Finally, in Chapter 6 we study uncertainties in execution time and energy consumption in data centers and their impact on several scheduling algorithms. We analyze real-world workloads and perform experimental evaluations to validate our energy consumption model. Results show the considered uncertainties significantly affect the accuracy of the scheduling algorithm and should be considered when designing scheduling algorithms in order to improve their accuracy.

7.2 Future work

There are a number of research challenges that need to be addressed in order to further advance with the work presented in this thesis. The main lines of future work are the following.

Address even larger problem instances. As computing demands are expected to keep increasing in the following few years, scheduling algorithms are expected to deal with larger workloads every year. It is important to study the scalability of the proposed algorithms and evaluate their accuracy when solving increasingly larger scenarios.

Compute lower bounds for each objective for small-sized instances. We propose to use relaxation techniques and exact methods to compute lower bounds for each objective when addressing small-sized instances of all problem formulations. Currently lower bounds have been computed for one problem formulation. Computing these lower bounds for all the problem formulations

will allow to study the behavior of the all proposed scheduling algorithms regarding optimality.

Integrate the proposed problem formulations. We propose to integrate the single data center problem formulation (proposed in Chapter 3) into the problem formulation considering a federation of heterogeneous data centers (proposed in Chapter 5). This will provide a more precise modeling for the low-level hierarchy scheduler by incorporating the cooling system among other characteristics. To accomplish this, the single data center problem formulation must be extended to consider heterogeneous computing resources and a computing load comprised by workflows of tasks.

Incorporate uncertainty into the problem formulation considering a federation of heterogeneous data centers. Incorporating the uncertainty model proposed in Chapter 6 into the problem formulation considering a federation of heterogeneous data centers (proposed in Chapter 5) will allow to study the robustness of the proposed scheduling algorithms when dealing with real-world uncertainties.

Consider other green energy sources and waste heat recycling. Finally, considering other green energy sources—such as wind energy—will allow to take into account off-site green energy generators. On top of that, considering waste heat recycling will allow to heat rooms during winter, improving the energy efficiency of the whole building hosting the data center.

Bibliography

Abbasi, Z., Pore, M., and Gupta, S. K. S. (2014). Impact of workload and renewable prediction on the value of geographical workload management. In Klingert, S., Hesselbach-Serra, X., Ortega, M. P., and Giuliani, G., editors, *Energy-Efficient Data Centers*, volume 8343 of *Lecture Notes in Computer Science*, pages 1–15. Springer, Berlin, Heidelberg.

Abraham, B. and Ledolter, J. (2008). *Seasonal Autoregressive Integrated Moving Average Models*, pages 281–321. John Wiley & Sons, Inc.

Abrishami, S., Naghibzadeh, M., and Epema, D. (2013). Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. *Future Generation Computer Systems*, 29(1):158–169.

Ahmad, I. and Ranka, S. (2012). *Handbook of Energy-Aware and Green Computing*. Chapman & Hall/CRC.

Al-Qawasmeh, A. M., Pasricha, S., Maciejewski, A. A., and Siegel, H. J. (2015). Power and thermal-aware workload allocation in heterogeneous data centers. *IEEE Transactions on Computers*, 64(2):477–491.

Alba, E. and Dorronsoro, B. (2008). *Cellular Genetic Algorithms*, volume 42 of *Operations Research/Computer Science Interfaces*. Springer-Verlag Heidelberg.

Ali, S., Maciejewski, A., Siegel, H., and Kim, J. (2004). Measuring the robustness of a resource allocation. *IEEE Transactions on Parallel and Distributed Systems*, 51(7):630–641.

Anastasopoulos, M., Tzanakaki, A., and Simeonidou, D. (2016). Stochastic energy efficient cloud service provisioning deploying renewable energy sources. *IEEE Journal on Selected Areas in Communications*, 34(12):3927–3940.

Andrae, A. S. and Edler, T. (2015). On global electricity usage of communication technology: trends to 2030. *Challenges*, 6(1):117–157.

ASHRAE Technical Committee (2011). Thermal guidelines for data processing environments 9.9. Technical report.

Bäck, T., Fogel, D., and Michalewicz, Z., editors (1997). *Handbook of evolutionary computation*. Oxford University Press.

Bader, J. and Zitzler, E. (2011). HypE : An algorithm for fast optimization. *Evolutionary Computation*, 19(1):45–76.

Bailey Lee, C., Schwartzman, Y., Hardy, J., and Snavely, A. (2005). Are user runtime estimates inherently inaccurate? In Feitelson, D. G., Rudolph, L., and Schwiegelshohn, U., editors, *Job Scheduling Strategies for Parallel Processing*, volume 3277 of *Lecture Notes in Computer Science*, pages 253–263. Springer, Berlin, Heidelberg.

Bansal, S., Kumar, P., and Singh, K. (2005). Dealing with heterogeneity through limited duplication for scheduling precedence constrained task graphs. *Journal of Parallel and Distributed Computing*, 65(4):479–491.

Barroso, L. and Hölzle, U. (2009). The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis Lectures on Computer Architecture*, 4(1):1–108.

Barroso, L. A., Clidaras, J., and Höelzle, U. (2013). *The Datacenter As a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers.

Barroso, L. A. and Hölzle, U. (2007). The case for energy-proportional computing. *Computer*, 40(12):33–37.

Baskiyar, S. and Abdel-Kader, R. (2010). Energy aware DAG scheduling on heterogeneous systems. *Cluster Computing*, 13(4):373–383.

Bender, M. A., Bunde, D. P., Demaine, E. D., Fekete, S. P., Leung, V. J., Meijer, H., and Phillips, C. A. (2008). Communication-aware processor allocation for supercomputers: Finding point sets of small average distance. *Algorithmica*, 50(2):279–298.

Bertsekas, D. P. and Tsitsiklis, J. N. (1989). *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Inc., NJ, USA.

Beume, N., Naujoks, B., and Emmerich, M. (2007). SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653 – 1669.

Bozdag, D., Ozguner, F., and Catalyurek, U. V. (2009). Compaction of schedules and a two-stage approach for duplication-based dag scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 20(6):857–871.

Braun, T., Siegel, H. J., Beck, N., Bölöni, L., Maheswaran, M., Reuther, A., Robertson, J., Theys, M., Yao, B., Hensgen, D., and Freund, R. (2001). A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, 61(6):810–837.

Brigham, O. (1974). *The Fast Fourier Transform*. Prentice-Hall, NJ, USA.

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., and Schulenburg, S. (2003). *Hyper-Heuristics: An Emerging Direction in Modern Search Technology*, volume 57 of *International Series in Operations Research & Management Science*, pages 457–474. Springer, Boston, MA.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., and Buyya, R. (2011). Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software-Practice & Experience*, 41(1):23–50.

Chalise, S., Golshani, A., Awasthi, S. R., Ma, S., Shrestha, B. R., Bajracharya, L., and Tonkoski, R. (2015). Data center energy systems: Current technology and future direction. In *IEEE Power & Energy Society General Meeting*, pages 1–5.

Chandra, G., Kapur, P., and Saraswat, K. C. (2002). Scaling trends for the on chip power dissipation. In *International Interconnect Technology Conference*, pages 170–172.

Chaudhry, M. T., Ling, T. C., Manzoor, A., Hussain, S. A., and Kim, J. (2015). Thermal-aware scheduling in green data centers. *ACM Computing Surveys*, 47(3):1–48.

Chen, S., Li, Z., Yang, B., and Rudolph, G. (2016a). Quantum-inspired hyper-heuristics for energy-aware scheduling on heterogeneous computing systems. *IEEE Transactions on Parallel and Distributed Systems*, 27(6):1796–1810.

Chen, T., Wang, X., and Giannakis, G. B. (2016b). Cooling-aware energy and workload management in data centers via stochastic optimization. *IEEE Journal on Selected Topics in Signal Processing*, 10(2):402–415.

Chen, T., Zhang, Y., Wang, X., and Giannakis, G. B. (2016c). Robust workload and energy management for sustainable data centers. *IEEE Journal on Selected Areas in Communications*, 34(3):651–664.

Chiong, R., Weise, T., and Michalewicz, Z. (2011). *Variants of Evolutionary Algorithms for Real-World Applications*. Springer, Berlin, Heidelberg.

Cirne, W. and Berman, F. (2001). A comprehensive model of the supercomputer workload. In *International Workshop on Workload Characterization*, pages 140–148.

Cisco Systems, Inc. (2016). Cisco global cloud index: Forecast and methodology, 2015–2020. Technical report.

Coello Coello, C. A., Lamont, G. B., and Van Veldhuizen, D. A. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Genetic and Evolutionary Computation. Springer, NJ, USA.

Coello Coello, C. A. and Lechuga, M. S. (2002). MOPSO: A proposal for multiple objective particle swarm optimization. *Congress on Evolutionary Computation*, 2:1051–1056.

Coello Coello, C. A. and Reyes Sierra, M. (2004). A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In Monroy, R., Arroyo-Figueroa, G., Sucar, L. E., and Sossa, H., editors, *Advances in Artificial Intelligence*, volume 2972 of *Lecture Notes in Computer Science*, pages 688–697. Springer, Berlin, Heidelberg.

Damousis, I. G., Alexiadis, M. C., Theocharis, J. B., and Dokopoulos, P. S. (2004). A fuzzy model for wind speed prediction and power generation in wind parks using spatial correlation. *IEEE Transactions on Energy Conversion*, 19(2):352–361.

Dayarathna, M., Wen, Y., and Fan, R. (2016). Data center energy consumption modeling: A survey. *IEEE Communications Surveys Tutorials*, 18(1):732–794.

de Assuncao, M. D., di Costanzo, A., and Buyya, R. (2009). Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. In *International Symposium on High Performance Distributed Computing*, pages 141–150.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., NY, USA.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Deng, W., Liu, F., Jin, H., Li, B., and Li, D. (2014). Harnessing renewable energy in cloud datacenters: Opportunities and challenges. *IEEE Network*, 28(1):48–55.

Dongarra, J. (1988). The LINPACK benchmark: An explanation. In *International Conference on Supercomputing*, pages 456–474.

Dorronsoro, B., Bouvry, P., Cañero, J., Maciejewski, A., and Siegel, H. (2010). Multi-objective robust static mapping of independent tasks on grids. In *IEEE Congress on Evolutionary Computation*, pages 3389–3396.

Dorronsoro, B., Nesmachnow, S., Taheri, J., Zomaya, A., Talbi, E.-G., and Bouvry, P. (2014a). A hierarchical approach for energy-efficient scheduling of large workloads in multicore distributed systems. *Sustainable Computing*, 4(4):252–261.

Dorronsoro, B., Ruiz, P., Danoy, G., Pigné, Y., and Bouvry, P. (2014b). *Evolutionary Algorithms for Mobile Ad Hoc Networks*. Wiley/IEEE Computer Society.

Dunn, O. J. (1961). Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64.

Durillo, J. and Nebro, A. (2011). jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771.

Ebrahimi, K., Jones, G. F., and Fleischer, A. S. (2014). A review of data center cooling technology, operating conditions and the corresponding low-grade waste heat recovery opportunities. *Renewable and Sustainable Energy Reviews*, 31:622 – 638.

Erol-Kantarci, M. and Mouftah, H. T. (2015). Energy-efficient information and communication infrastructures in the smart grid: A survey on interactions and open issues. *IEEE Communications Surveys & Tutorials*, 17(1):179–197.

Feitelson, D. G., Rudolph, L., and Schwiegelshohn, U. (2005). Parallel job scheduling — A status report. In Feitelson, D. G., Rudolph, L., and Schwiegelshohn, U., editors, *Job Scheduling Strategies for Parallel Processing*, volume 3277 of *Lecture Notes in Computer Science*, pages 1–16. Springer, Berlin, Heidelberg.

Fulpagare, Y. and Bhargav, A. (2015). Advances in data center thermal management. *Renewable and Sustainable Energy Reviews*, 43:981 – 996.

Garg, R. and Singh, A. K. (2016). Energy-aware workflow scheduling in grid under QoS constraints. *Arabian Journal for Science and Engineering*, 41(2):495–511.

Ghafoor, A. and Yang, J. (1993). Distributed heterogeneous supercomputing management system. *Computer*, 26(6):78–86.

Ghamkhari, M. and Mohsenian-Rad, H. (2013). Energy and performance management of green data centers: A profit maximization approach. *IEEE Transactions on Smart Grid*, 4(2):1017–1025.

Goiri, Í., Haque, M. E., Le, K., Beauchea, R., Nguyen, T. D., Guitart, J., Torres, J., and Bianchini, R. (2015a). Matching renewable energy supply and demand in green datacenters. *Ad Hoc Networks*, 25:520–534.

Goiri, I., Katsak, W., Le, K., Nguyen, T., and Bianchini, R. (2013). Parasol and GreenSwitch: managing datacenters powered by renewable energy. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 51–64.

Goiri, I., Katsak, W., Le, K., Nguyen, T., and Bianchini, R. (2014). Designing and managing data centers powered by renewable energy. *IEEE Micro*, 34(3):8–16.

Goiri, Í., Nguyen, T. D., and Bianchini, R. (2015b). CoolAir: Temperature- and variation-aware management for free-cooled datacenters. *ACM SIGPLAN Notices*, 50(4):253–265.

Goudarzi, H. and Pedram, M. (2016). Hierarchical SLA-driven resource management for peak power-aware and energy-efficient operation of a cloud datacenter. *IEEE Transactions on Cloud Computing*, 4(2):222–236.

Guo, Y. and Fang, Y. (2013). Electricity cost saving strategy in data centers by using energy storage. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1149–1160.

Guo-ning, G., Ting-Iei, H., and Shuai, G. (2010). Genetic simulated annealing algorithm for task scheduling based on cloud computing environment. In *International Conference on Intelligent Computing and Integrated Systems*, pages 60–63.

Guzek, M., Bouvry, P., and Talbi, E. G. (2015). A survey of evolutionary computation for resource management of processing in cloud computing. *IEEE Computational Intelligence Magazine*, 10(2):53–67.

Habibi Khalaj, A., Scherer, T., Siriwardana, J., and Halgamuge, S. K. (2015). Multi-objective efficiency enhancement using workload spreading in an operational data center. *Applied Energy*, 138:432–444.

Hagras, T. and Janecek, J. (2005). A high performance, low complexity algorithm for compile-time task scheduling in heterogeneous systems. *Parallel Computing*, 31(7):653 – 670.

Haywood, A. M., Sherbeck, J., Phelan, P., Varsamopoulos, G., and Gupta, S. K. S. (2015). The relationship among CPU utilization, temperature, and thermal power for waste heat utilization. *Energy Conversion and Management*, 95:297–303.

Herrero, J. M., Reynoso-Meza, G., Martínez, M., Blasco, X., and Sanchis, J. (2014). A smart-distributed pareto front using the ev-moga evolutionary algorithm. *International Journal on Artificial Intelligence Tools*, 23(2):1–22.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):pp. 65–70.

Huang, Q., Su, S., Li, J., Xu, P., Shuang, K., and Huang, X. (2012). Enhanced energy-efficient scheduling for parallel applications in cloud. *International Symposium on Cluster, Cloud and Grid Computing*, pages 781–786.

Hunter, J. S. (1986). The exponentially weighted moving average. *Journal of Quality Technology*, 18(4):203–210.

Ibarra, O. H. and Kim, C. E. (1977). Heuristic algorithms for scheduling independent tasks on nonidentical processors. *Journal of the ACM*, 24(2):280–289.

Ishihara, T. and Yasuura, H. (1998). Voltage scheduling problem for dynamically variable voltage processors. In *International Symposium on Low Power Electronics and Design*, number February, pages 197–202.

Iturriaga, S., Dorronsoro, B., and Nesmachnow, S. (2017). Multiobjective evolutionary algorithms for energy and service level scheduling in a federation of distributed datacenters. *International Transactions in Operational Research*, 24(1-2):199–228.

Iturriaga, S., García, S., and Nesmachnow, S. (2014). An empirical study of the robustness of energy-aware schedulers for high performance computing systems under uncertainty. In Hernández, G., Barrios Hernández, C. J., Díaz, G., García Garino, C., Nesmachnow, S., Pérez-Acle, T., Storti, M., and Vázquez, M., editors, *High Performance Computing*, volume 485 of *Communications in Computer and Information Science*, pages 143–157. Springer, Berlin, Heidelberg.

Iturriaga, S. and Nesmachnow, S. (2015). Multiobjective scheduling of green-powered datacenters considering QoS and budget objectives. In *Innovative Smart Grid Technologies Latin America*, pages 570–573.

Iturriaga, S. and Nesmachnow, S. (2016). Scheduling energy efficient data centers using renewable energy. *Electronics*, 5(4).

Iturriaga, S., Nesmachnow, S., Dorronsoro, B., and Bouvry, P. (2013). Energy efficient scheduling in heterogeneous systems with a parallel multiobjective local search. *Computing and Informatics Journal*, 32(2):273–294.

Iturriaga, S., Nesmachnow, S., Tchernykh, A., and Dorronsoro, B. (2016). Multiobjective workflow scheduling in a federation of heterogeneous green-powered data centers. In *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 596–599.

Jayasinghe, D., Pu, C., Eilam, T., Steinder, M., Whally, I., and Snible, E. (2011). Improving performance and availability of services hosted on iaas clouds with structural constraint-aware virtual machine placement. In *International Conference on Services Computing*, pages 72–79.

Jena, R. K. (2015). Multi objective task scheduling in cloud environment using nested PSO framework. *Procedia Computer Science*, 57:1219–1227.

Kafil, M. and Ahmad, I. (1998). Optimal task assignment in heterogeneous distributed computing systems. *IEEE Concurrency*, 6(3):42–51.

Katrinis, K. M. and Tzanakaki, A. (2011). On the dimensioning of WDM optical networks with impairment-aware regeneration. *IEEE/ACM Transactions on Networking*, 19(3):735–746.

Kaushik, A. and Vidyarthi, D. P. (2016). An energy-efficient reliable grid scheduling model using NSGA-II. *Engineering with Computers*, 32(3):355–376.

Kessaci, Y., Melab, N., and Talbi, E. G. (2013). A Pareto-based metaheuristic for scheduling HPC applications on a geographically distributed cloud federation. *Cluster Computing*, 16(3):451–468.

Khan, S. and Ahmad, I. (2009). A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids. *IEEE Transactions on Parallel and Distributed Systems*, 20(3):346–360.

Khan, S. U. and Zomaya, A. Y. (2015). *Handbook on data centers*. Springer, NY, USA.

Kiani, A. and Ansari, N. (2016). Profit maximization for geographical dispersed green data centers. *IEEE Transactions on Smart Grid*, pages 1–16. To appear.

Kim, J.-Y., Chang, H.-J., Jung, Y.-H., Cho, K.-M., and Augenbroe, G. (2017). Energy conservation effects of a multi-stage outdoor air enabled cooling system in a data center. *Energy and Buildings*, 138:257–270.

Kim, K., Buyya, R., and Kim, J. (2007). Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters. In *International Symposium on Cluster Computing and the Grid*, pages 541–548.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Kleywegt, A. J., Shapiro, A., and Homem-de-Mello, T. (2002). The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502.

Knowles, J., Thiele, L., and Zitzler, E. (2006). A tutorial on the performance assessment of stochastic multiobjective optimizers. Report 214, Computer Engineering and Networks Laboratory, ETH Zurich.

Koomey, J. (2008). Worldwide electricity used in data centers. *Environmental Research Letters*, 3(3):1–8.

Koomey, J. (2011). Growth in data center electricity use 2005–2010. Technical report, Analytic Press.

Kruskal, W. H. and Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621.

Kumar, P. and Verma, A. (2012). Independent task scheduling in cloud computing by improved genetic algorithm. *International Journal of Advanced Research in Computer Science and Software Engineering Research*, 2(5):5–8.

Kumar, S., Sabharwal, Y., Garg, R., and Heidelberger, P. (2008). Optimization of all-to-all communication on the blue gene/L supercomputer. *International Conference on Parallel Processing*, pages 320–329.

Le, K., Bianchini, R., Nguyen, T. D., Bilgir, O., and Martonosi, M. (2010). Capping the brown energy consumption of internet services at low cost. *International Conference on Green Computing*, pages 3–14.

Lee, E. K., Viswanathan, H., and Pompili, D. (2012). VMAP: Proactive thermal-aware virtual machine allocation in HPC cloud datacenters. In *International Conference on High Performance Computing*, pages 1–10.

Lee, E. K., Viswanathan, H., and Pompili, D. (2017). Proactive thermal-aware resource management in virtualized HPC cloud datacenters. *IEEE Transactions on Cloud Computing*, 5(2):234–248.

Lee, Y. and Zomaya, A. (2009). Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling. In *International Symposium on Cluster Computing and the Grid*, pages 92–99.

Lee, Y. and Zomaya, A. (2011). Energy conscious scheduling for distributed computing systems under different operating conditions. *IEEE Transactions on Parallel and Distributed Systems*, 22(8):1374–1381.

Lei, H., Wang, R., Zhang, T., Liu, Y., and Zha, Y. (2016). A multi-objective co-evolutionary algorithm for energy-efficient scheduling on a green data center. *Computers & Operations Research*, 75:103–117.

Lei, H., Zhang, T., Liu, Y., Zha, Y., and Zhu, X. (2015). SGEESS: Smart green energy-efficient scheduling strategy with dynamic electricity price for data center. *Journal of Systems and Software*, 108:23–38.

Leung, J., Kelly, L., and Anderson, J. H. (2004). *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press, Inc., FL, USA.

Leverich, J., Monchiero, M., Talwar, V., Ranganathan, P., and Kozyrakis, C. (2010). Power management of datacenter workloads using per-core power gating. *IEEE Computer Architecture Letters*, 8(2):48–51.

Li, Y., Liu, Y., and Qian, D. (2009). A heuristic energy-aware scheduling algorithm for heterogeneous clusters. In *International Conference on Parallel and Distributed Systems*, pages 407–413.

Lifka, D. A. (1995). The ANL/IBM SP scheduling system. In Feitelson, D. G. and Rudolph, L., editors, *Job Scheduling Strategies for Parallel Processing*, volume 949 of *Lecture Notes in Computer Science*, pages 295–303. Springer, Berlin, Heidelberg.

Lin, M., Wierman, A., Andrew, L. L. H., and Thereska, E. (2013). Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Transactions on Networking*, 21(5):1378–1391.

Lin, R. and Deng, Y. (2017). Allocating workload to minimize the power consumption of data centers. *Frontiers of Computer Science*, 11(1):105–118.

Lindberg, P., Leingang, J., Lysaker, D., Khan, S., and Li, J. (2012). Comparison and analysis of eight scheduling heuristics for the optimization of energy consumption and makespan in large-scale distributed systems. *The Journal of Supercomputing*, 59(1):323–360.

Liu, Z., Lin, M., Wierman, A., Low, S. H., and Andrew, L. L. H. (2011). Geographical load balancing with renewables. *ACM SIGMETRICS Performance Evaluation Review*, 39(3):62–66.

Lopes, R. V. and Menasce, D. (2016). A taxonomy of job scheduling on distributed computing systems. *IEEE Transactions on Parallel and Distributed Systems*, 27(12):3412–3428.

Manousakis, I., Goiri, I. n., Sankar, S., Nguyen, T. D., and Bianchini, R. (2015). CoolProvision: Underprovisioning datacenter cooling. In *ACM Symposium on Cloud Computing*, pages 356–367.

Martínez-Iranzo, M., Herrero, J. M., Sanchis, J., Blasco, X., and García-Nieto, S. (2009). Applied pareto multi-objective optimization by stochastic solvers. *Engineering Applications of Artificial Intelligence*, 22(3):455 – 465.

Mehta, A. M., Smith, J., Siegel, H. J., Maciejewski, A. A., Jayaseelan, A., and Ye, B. (2007). Dynamic resource allocation heuristics that manage tradeoff between makespan and robustness. *Journal of Supercomputing*, 42(1):33–58.

Mei, J., Li, K., and Li, K. (2013). Energy-aware task scheduling in heterogeneous computing environments. *Cluster Computing*, 17(2):537–550.

Meisner, D., Gold, B. T., and Wenisch, T. F. (2009). PowerNap: eliminating server idle power. *ACM SIGARCH Computer Architecture News*, 37(1):205–216.

Meng, J., McCauley, S., Kaplan, F., Leung, V. J., and Coskun, A. K. (2015). Simulation and optimization of HPC job allocation for jointly reducing communication and cooling costs. *Sustainable Computing: Informatics and Systems*, 6:48–57.

Mezmaz, M., Melab, N., Kessaci, Y., Lee, Y., Talbi, E., Zomaya, A., and Tuyttens, D. (2011). A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. *Journal of Parallel Distributed Computing*, 71:1497–1508.

Minh, T. N. and Wolters, L. (2010). Using historical data to predict application runtimes on backfilling parallel systems. In *Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, pages 246–252.

Moon, H., Chi, Y., and Hacigümüş, H. (2011). Performance evaluation of scheduling algorithms for database services with soft and hard SLAs. In *International Workshop on Data Intensive Computing in the Clouds*, pages 81–90.

Moore, J., Chase, J., Ranganathan, P., and Sharma, R. (2005). Making scheduling "cool": Temperature-aware workload placement in data centers. In *USENIX Annual Technical Conference*, pages 61–74.

Mu'alem, A. and Feitelson, D. (2001). Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. *IEEE Transactions on Parallel and Distributed Systems*, 12(6):529–543.

Mukherjee, T., Banerjee, A., Varsamopoulos, G., Gupta, S. K. S., and Rungta, S. (2009). Spatio-temporal thermal-aware job scheduling to minimize energy consumption in virtualized heterogeneous data centers. *Computer Networks*, 53(17):2888–2904.

Nebro, A., Durillo, J., Luna, F., Dorronsoro, B., and Alba, E. (2009). MOCell: A cellular genetic algorithm for multiobjective optimization. *International Journal of Intelligent Systems*, 24(7):726–746.

Neely, M. J. (2010). *Stochastic Network Optimization with Application to Communication and Queueing Systems*, volume 3 of *Synthesis Lectures on Communication Networks*. Morgan and Claypool Publishers.

Nesmachnow, S. (2014). An overview of metaheuristics: Accurate and efficient methods for optimisation. *International Journal of Metaheuristics*, 3(4):320–347.

Nesmachnow, S., Dorronsoro, B., and Bouvry, P. (2014a). Energy-aware workflow scheduling in datacenters. In *ALIO/EURO Workshop on Applied Combinatorial Optimization*, pages 1–6.

Nesmachnow, S., Dorronsoro, B., Pecero, J. E., and Bouvry, P. (2013). Energy-aware scheduling on multicore heterogeneous grid computing systems. *Journal of Grid Computing*, 11(4):653–680.

Nesmachnow, S., Perfumo, C., and Goiri, I. (2014b). Controlling datacenter power consumption while maintaining temperature and QoS levels. In *IEEE International Conference on Cloud Networking*, pages 242–247.

Nesmachnow, S., Perfumo, C., and Goiri, Í. (2015). Holistic multiobjective planning of datacenters powered by renewable energy. *Cluster Computing*, 18(4):1379–1397.

Oró, E., Depoorter, V., Garcia, A., and Salom, J. (2015a). Energy efficiency and renewable energy integration in data centres. Strategies and modelling review. *Renewable and Sustainable Energy Reviews*, 42:429–445.

Oró, E., Depoorter, V., Pflugradt, N., and Salom, J. (2015b). Overview of direct air free cooling and thermal energy storage potential energy savings in data centres. *Applied Thermal Engineering*, 85:100–110.

Parnell, L. A., Demetriou, D. W., and Zhang, E. Y. (2016). Combining cooling technology and facility design to improve HPC data center energy efficiency. In *2016 15th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, pages 417–425. IEEE.

Patel, C. D., Sharma, R. K., Bash, C. E., and Beitelmal, M. H. (2006). Energy flow in the information technology stack: Introducing the coefficient of performance of the ensemble. In *ASME International Mechanical Engineering Congress and Exposition*, pages 233–241.

Paul, D., Zhong, W. D., and Bose, S. K. (2016). Energy efficiency aware load distribution and electricity cost volatility control for cloud service providers. *Journal of Network and Computer Applications*, 59:185–197.

Pecero, J. E., Bouvry, P., Huacuja, H. J. F., and Khan, S. U. (2011). A multi-objective GRASP algorithm for joint optimization of energy consumption and schedule length of precedence-constrained applications. In *IEEE International Conference on Dependable, Autonomic and Secure Computing*, pages 510–517.

Peng, Y., Kang, D. K., Al-Hazemi, F., and Youn, C. H. (2017). Energy and QoS aware resource allocation for heterogeneous sustainable cloud datacenters. *Optical Switching and Networking*, 23:225–240.

Pinel, F., Dorronsoro, B., Pecero, J., Bouvry, P., and Khan, S. (2013). A two-phase heuristic for the energy-efficient scheduling of independent tasks on computational grids. *Cluster Computing*, 16(3):421–433.

Polverini, M., Cianfrani, A., Ren, S., and Vasilakos, A. V. (2014). Thermal-aware scheduling of batch jobs in geographically distributed data centers. *IEEE Transactions on Cloud Computing*, 2(1):71–84.

Pore, M., Abbasi, Z., Gupta, S. K. S., and Varsamopoulos, G. (2015). Techniques to achieve energy proportionality in data centers: A survey. In Khan, S. U. and Zomaya, A. Y., editors, *Handbook on Data Centers*, pages 109–162. Springer, New York, NY.

Prajapati, H. B. and Shah, V. A. (2014). Bandwidth-aware scheduling of workflow application on multiple grid sites. *Journal of Computer Networks and Communications*, 2014:1–15.

Quezada-Pina, A., Tchernykh, A., González-García, J., Hirales-Carbajal, A., Ramírez-Alcaraz, J., Schwiegelshohn, U., Yahyapour, R., and Miranda-López, V. (2012). Adaptive parallel job scheduling with resource admissible allocation on two-level hierarchical grids. *Future Generation Computer Systems*, 28(7):965–976.

Qureshi, A., Weber, R., Balakrishnan, H., Guttag, J., and Maggs, B. (2009). Cutting the electric bill for internet-scale systems. *ACM SIGCOMM Computer Communication Review*, 39(4):123.

Rahman, A., Liu, X., and Kong, F. (2014). A survey on geographic load balancing based data center power management in the smart grid environment. *IEEE Communications Surveys & Tutorials*, 16(1):214–233.

Rajabi, A., Faragardi, H. R., and Nolte, T. (2014). An efficient scheduling of hpc applications on geographically distributed cloud data centers. In Jahangir, A. H., Movaghar, A., and Asadi, H., editors, *Computer Networks and Distributed Systems*, volume 428 of *Communications in Computer and Information Science*, pages 155–167. Springer International Publishing.

Rao, L., Liu, X., Ilic, M. D., and Liu, J. (2012a). Distributed coordination of internet data centers under multiregional electricity markets. *Proceedings of the IEEE*, 100(1):269–282.

Rao, L., Liu, X., Xie, L., and Liu, W. (2010). Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment. *IEEE International Conference on Computer Communications*, pages 1–9.

Rao, L., Liu, X., Xie, L., and Liu, W. (2012b). Coordinated energy cost management of distributed internet data centers in smart grid. *IEEE Transactions on Smart Grid*, 3(1):50–58.

Ren, S., He, Y., and Xu, F. (2012). Provably-efficient job scheduling for energy and fairness in geographically distributed data centers. *International Conference on Distributed Computing Systems*, pages 22–31.

Ren, Z., Zhang, X., and Shi, W. (2015). Resource scheduling in data-centric systems. In Khan, S. U. and Zomaya, A. Y., editors, *Handbook on Data Centers*, pages 1307–1330. Springer, NY, USA.

Riquelme, N., Lücken, C. V., and Baran, B. (2015). Performance metrics in multi-objective optimization. In *Latin American Computing Conference*, pages 1–11.

Rizvandi, N., Taheri, J., and Zomaya, A. (2011). Some observations on optimal frequency selection in DVFS-based energy consumption minimization. *Journal Parallel Distributed Computing*, 71(8):1154–1164.

Rodriguez, M. A. and Buyya, R. (2017). A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments. *Concurrency and Computation: Practice and Experience*, 29(8):1–23.

Rong, H., Zhang, H., Xiao, S., Li, C., and Hu, C. (2016). Optimizing energy consumption for data centers. *Renewable and Sustainable Energy Reviews*, 58:674–691.

Sajid, M. and Raza, Z. (2017). Energy-aware stochastic scheduler for batch of precedence-constrained jobs on heterogeneous computing system. *Energy*, 125:258–274.

Schwiegelshohn, U. and Tchernykh, A. (2012). Online scheduling for cloud computing and different service levels. In *IEEE International Parallel and Distributed Processing Symposium Workshops PhD Forum*, pages 1067–1074.

Sharifi, M., Shahrivari, S., and Salimi, H. (2013). PASTA: a power-aware solution to scheduling of precedence-constrained tasks on heterogeneous computing resources. *Computing*, 95(1):67–88.

Shehabi, A., Smith, S., Horner, N., Azevedo, I., Brown, R., Koomey, J., Masanet, E., Sartor, D., Herrlin, M., and Lintner, W. (2016). United states data center energy usage report. Technical Report LBNL-1005775, Lawrence Berkeley National Laboratory, Berkeley, California.

Shi, L., Zhang, Z., and Robertazzi, T. (2017). Energy-aware scheduling of embarrassingly parallel jobs and resource allocation in cloud. *IEEE Transactions on Parallel and Distributed Systems*, 28(6):1607–1620.

Shmueli, E. and Feitelson, D. (2009). On simulation and design of parallel-systems schedulers: Are we doing the right thing? *IEEE Transactions on Parallel and Distributed Systems*, 20(7):983–996.

Shuja, J., Bilal, K., Madani, S. A., Othman, M., Ranjan, R., Balaji, P., and Khan, S. U. (2016a). Survey of techniques and architectures for designing energy-efficient data centers. *IEEE Systems Journal*, 10(2):507–519.

Shuja, J., Gani, A., Shamshirband, S., Ahmad, R. W., and Bilal, K. (2016b). Sustainable cloud data centers: A survey of enabling techniques and technologies. *Renewable and Sustainable Energy Reviews*, 62:195–214.

Siegel, S. (1956). *Nonparametric statistics for the behavioral sciences*. McGraw-Hill.

Song, Z., Zhang, X., and Eriksson, C. (2015). Data center energy and cost saving evaluation. *Energy Procedia*, 75:1255–1260.

Sprent, P. and Smeeton, N. (2000). *Applied Nonparametric Statistical Methods*. Texts in Statistical Science. Chapman & Hall/CRC.

Sun, X., Ansari, N., and Wang, R. (2016). Optimizing resource utilization of a data center. *IEEE Communications Surveys & Tutorials*, 18(4):2822–2846.

Taheri, J., Zomaya, A., and Khan, S. (2013). Grid simulation tools for job scheduling and datafile replication. In Khan, S. U., Zomaya, A. Y., and Lizhe, W., editors, *Scalable Computing and Communications: Theory and Practice*, chapter 35, pages 777–797. John Wiley & Sons, Inc., NJ, USA.

Tang, Q., Gupta, S. K. S., and Varsamopoulos, G. (2007). Thermal-aware task scheduling for data centers through minimizing heat recirculation. In *IEEE International Conference on Cluster Computing*, pages 129–138.

Tang, Q., Gupta, S. K. S., and Varsamopoulos, G. (2008). Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach. *IEEE Transactions on Parallel and Distributed Systems*, 19(11):1458–1472.

Tang, Q., Mukherjee, T., Gupta, S. K. S., and Cayton, P. (2006). Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters. In *International Conference on Intelligent Sensing and Information Processing*, pages 203–208.

Tang, W., Desai, N., Buettner, D., and Lan, Z. (2013). Job scheduling with adjusted runtime estimates on production supercomputers. *Journal of Parallel and Distributed Computing*, 73(7):926 – 938.

Tang, X., Li, K., Liao, G., Fang, K., and Wu, F. (2011). A stochastic scheduling algorithm for precedence constrained tasks on Grid. *Future Generation Computer Systems*, 27(8):1083–1091.

Tang, Z., Qi, L., Cheng, Z., Li, K., Khan, S. U., and Li, K. (2016). An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment. *Journal of Grid Computing*, 14(1):55–74.

Tchernykh, A., Lozano, L., Schwiegelshohn, U., Bouvry, P., Pecero, J., Nesmachnow, S., and Drozdov, A. (2015). Online bi-objective scheduling for IaaS clouds ensuring quality of service. *Journal of Grid Computing*, 14(1):5–22.

Toosi, A. N., Qu, C., de Assunção, M. D., and Buyya, R. (2017). Renewable-aware geographical load balancing of web applications for sustainable data centers. *Journal of Network and Computer Applications*, 83:155 – 168.

Topcuoglu, H., Hariri, S., and Wu, M.-Y. W. M.-Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(3):260–274.

Tsafrir, D. (2010). Using inaccurate estimates accurately. In Frachtenberg, E. and Schwiegelshohn, U., editors, *Job Scheduling Strategies for Parallel Processing*, volume 6253 of *Lecture Notes in Computer Science*, pages 208–221. Springer, Berlin, Heidelberg.

Valentini, G., Lassonde, W., Khan, S., Min-Allah, N., Madani, S., Li, J., Zhang, L., Wang, L., Ghani, N., Kolodziej, J., Li, H., Zomaya, A., Xu, C.-Z., Balaji, P., Vishnu, A., Pinel, F., Pecero, J., Kliazovich, D., and Bouvry, P. (2013). An overview of energy efficiency techniques in cluster computing systems. *Cluster Computing*, 16(1):3–15.

Van Heddeghem, W., Lambert, S., Lannoo, B., Colle, D., Pickavet, M., and Demeester, P. (2014). Trends in worldwide ict electricity consumption from 2007 to 2012. *Computer Communications*, 50:64 – 76.

Verma, A., Ahuja, P., and Neogi, A. (2008). pMapper: Power and migration cost aware application placement in virtualized systems. In Issarny, V. and Schantz, R., editors, *Middleware*, volume 5346 of *Lecture Notes in Computer Science*, pages 243–264. Springer, Berlin, Heidelberg.

Von Kistowski, J., Beckett, J., Lange, K. D., Block, H., Arnold, J. A., and Kounev, S. (2015). Energy efficiency of hierarchical server load distribution strategies. In *IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, pages 75–84.

Wang, R., Purshouse, R. C., and Fleming, P. J. (2013). Preference-inspired coevolutionary algorithms for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 17(4):474–494.

Wang, S., Liu, J., Chen, J. J., and Liu, X. (2011). PowerSleep: A smart power-saving scheme with sleep for servers under response time constraint. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 1(3):289–298.

Wang, Y.-R., Huang, K.-C., and Wang, F.-J. (2016). Scheduling online mixed-parallel workflows of rigid tasks in heterogeneous multi-cluster environments. *Future Generation Computer Systems*, 60:35–47.

Wang, Z., Tolia, N., and Bash, C. (2010). Opportunities and challenges to unify workload, power, and cooling management in data centers. *ACM SIGOPS Operating Systems Review*, 44(3):41.

Weiser, M., Welch, B., Demers, A., and Shenker, S. (1996). *Scheduling for Reduced CPU Energy*, volume 353 of *The International Series in Engineering and Computer Science*, pages 449–471. Springer, Boston, MA.

Wellons, J., Dai, L., Xue, Y., and Cui, Y. (2010). Augmenting predictive with oblivious routing for wireless mesh networks under traffic uncertainty. *Computer Networks*, 54(2):178–195.

Wierman, A., Andrew, L. L. H., and Tang, A. (2012). Power-aware speed scaling in processor sharing systems: Optimality and robustness. *Performance Evaluation*, 69(12):601–622.

Wu, F., Wu, Q., and Tan, Y. (2015). Workflow scheduling in cloud: a survey. *The Journal of Supercomputing*, 71(9):3373–3418.

Wu, F., Wu, Q., Tan, Y., Li, R., and Wang, W. (2016). PCP-B2: Partial critical path budget balanced scheduling algorithms for scientific workflow applications. *Future Generation Computer Systems*, 60:22–34.

Xie, G., Xiao, X., Li, R., and Li, K. (2016). Schedule length minimization of parallel applications with energy consumption constraints using heuristics on heterogeneous distributed systems. *Concurrency and Computation: Practice and Experience*, pages 1–10.

Xu, H., Feng, C., and Li, B. (2015). Temperature aware workload managementin geo-distributed data centers. *IEEE Transactions on Parallel and Distributed Systems*, 26(6):1743–1753.

Yu, L., Jiang, T., and Cao, Y. (2015). Energy cost minimization for distributed internet data centers in smart microgrids considering power outages. *IEEE Transactions on Parallel and Distributed Systems*, 26(1):120–130.

Yu, X. and Gen, M. (2012). *Introduction to Evolutionary Algorithms*. Springer, London, UK.

Zachary Woodruff, J., Brenner, P., Buccellato, A. P. C., and Go, D. B. (2014). Environmentally opportunistic computing: A distributed waste heat reutilization approach to energy-efficient buildings and data centers. *Energy and Buildings*, 69:41–50.

Zhang, H., Shao, S., Xu, H., Zou, H., and Tian, C. (2014). Free cooling of data centers: A review. *Renewable and Sustainable Energy Reviews*, 35:171–182.

Zhang, W., Wen, Y., Wong, Y., Toh, K., and Chen, C.-H. (2016). Towards joint optimization over ICT and cooling systems in data centre: A survey. *IEEE Communications Surveys & Tutorials*, 18(3):1596–1616.

Zhang, Y., Gatsis, N., and Giannakis, G. B. (2013). Robust energy management for microgrids with high-penetration renewables. *IEEE Transactions on Sustainable Energy*, 4(4):944–953.

Zhu, D., Melhem, R., and Childers, B. (2003). Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems. *IEEE Transaction on Parallel and Distributed Systems*, 14(7):686–700.

Zhu, Z., Zhang, G., Li, M., and Liu, X. (2016). Evolutionary multi-objective workflow scheduling in cloud. *IEEE Transactions on Parallel and Distributed Systems*, 27(5):1344–1357.

Zitzler, E. and Künzli, S. (2004). Indicator-based selection in multiobjective search. In Yao, X., Burke, E. K., Lozano, J. A., Smith, J., Merelo-Guervós, J. J., Bullinaria, J. A., Rowe, J. E., Tiňo, P., Kabán, A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature*, volume 3242 of *Lecture Notes in Computer Science*, pages 832–842. Springer, Berlin, Heidelberg.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the strength pareto evolutionary algorithm. Report 103, Computer Engineering and Networks Laboratory, ETH Zurich.

Zitzler, E. and Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms - a comparative case study. In *International Conference on Parallel Problem Solving from Nature*, pages 292–304, London, UK. Springer.

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transaction on Evolutionary Computation*, 3(4):257–271.

Zomaya, A. and Lee, Y. (2012). *Energy Efficient Distributed Computing Systems*. Wiley-IEEE Computer Society Press.

Zong, Z., Manzanares, A., Ruan, X., and Qin, X. (2011). EAD and PEBD: Two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters. *IEEE Transactions on Computers*, 60(3):360–374.

Artwork: Icons included in this thesis were created by Gan Khoon Lay, Sarah Joy, Steve Morris, Arthur Shlain, Adrien Coquet, Ron Scott, Maurizio Fusilo, and Erick Miranda Vazquez from the Noun Project and are freely available under Creative Commons license.